Big Data Software

Spring 2017

Bloomington, Indiana

Editor:
Gregor von Laszewski
Department of Intelligent Systems
Engeneering
Indiana University
laszewski@gmail.com

Contents

Apache Ranger

AVADHOOT AGASTI[1,*,+]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: aagasti@indiana.edu
[+] HID - SL-IO-3000

paper2, March 25, 2017

Apache Hadoop provides various data storage, data access and data processing services. Apache Ranger is part of the Hadoop eco-system. Apache Ranger provides capability to perform security administration tasks for storage, access and processing of data in Hadoop. Using Ranger, Hadoop administrator can perform security administration tasks using a central UI or Restful web services. He can define policies which enable users/user-groups to perform specific action using Hadoop components and tools. Ranger provides role based access control for datasets on Hadoop at column and row level. Ranger also provides centralized auditing of user access and security related administrative actions.

Keywords: Apache Ranger, LDAP, Active Directory, Apache Knox, Apache Atlas, Apache Hive, Apache Hadoop, Yarn, Apache HBase, Apache Storm, Apache Kafka, Data Lake, Apache Sentry

https://github.com/avadhoot-agasti/sp17-i524/tree/master/paper2/S17-IO-3000/report.pdf

1. INTRODUCTION

Apache Ranger is open source software project designed to provide centralized security services to various components of Apache Hadoop. Apache Hadoop provides various mechanism to store, process and access the data. Each Apache tool has its own security mechanism. This increases administrative overhead and is also error prone. Apache Ranger fills this gap to provide a central security and auditing mechanism for various Hadoop components. Using Ranger, Hadoop administrator can perform security administration tasks using a central UI or Restful web services. He can define policies which enable users/user-groups to perform specific action using Hadoop components and tools. Ranger provides role based access control for datasets on Hadoop at column and row level. Ranger also provides centralized auditing of user access and security related administrative actions.

2. ARCHITECTURE OVERVIEW

[1] describes the important components of Ranger as explained below:

2.1. Ranger Admin Portal

Ranger Admin Portal is the main interaction point for the user. Using Admin Portal, user can define policies. The policies are stored in Policy Database. The Policies are polled by various plugins. The Admin Portal also collects the audit data from plugins and stores in HDFS or in a relational database.

2.2. Ranger Plugins

Plugins are Java Programs, which are invoked as part of the cluster component. For example, the ranger-hive plugin is embedded as part of Hive Server2. The plugins cache the policies, and intercept the user request and evaluates it against the policies. Plugins also collect the audit data for that specific component and send to Admin Portal.

2.3. User group sync

While Ranger provides authorization or access control mechanism, it needs to know the users and the groups. Ranger integrates with Unix users management or LDAP or Active Directory to fetch the users and groups. The User group sync component is responsible for this integration.

3. HADOOP COMPONENTS SUPPORTED BY RANGER

Ranger supports auditing and authorization for following Hadoop components [2].

3.1. Apache Hadoop and HDFS

Apache Ranger provides plugin for Hadoop, which helps in enforcing data access policies. The HDFS plugin works with Name Node to check if the user's access request to a file on HDFS is valid or not.

3.2. Apache Hive

Apache Hive provides SQL interface on top of the data stored in HDFS. Apache Hive supports two types of authorization -

Storage based authorization and SQL standard authorization. Ranger provides centralized authorization interface for Hive which provides granular access control at table and column level. Ranger implements a plugin which is part of Hive Server2.

3.3. Apache HBase

Apache HBase is NoSQL database implemented on top of Hadoop and HDFS. Ranger provides coprocessor plugin for HBase, which performs authorization checks and audit log collections.

3.4. Apache Storm

Ranger provides plugin to Nimbus server which helps in performing the security authorization on Apache Storm.

3.5. Apache Knox

Apache Knox provides service level authorization for users and groups. Ranger provides plugin for Knox using which, administration of policies can be supported. The audit over Knox data enables user to perform detailed analysis of who and when accessed Knox.

3.6. Apache Solr

Solr provides free text search capabilities on top of Hadoop. Ranger is useful to protect Solr collections from unauthorized usage.

3.7. Apache kafka

Ranger can manager access control on Kafka topics. Policies can be implemented to control which users can write to a Kafka topic and which users can read from a Kafka topic.

3.8. Yarn

Yarn is resource management layer for Hadoop. Administrators can setup queues in Yarn and then allocate users and resources per queue basis. Policies can be defined in Ranger to define who can write to various Yarn queues.

4. IMPORTANT FEATURES OF RANGER

The blog article [3] explains the 2 important features of Apache Ranger.

4.1. Dynamic Column Masking

Dynamic data masking at column level is an important feature of Apache Ranger . Using this feature, the administrator can setup data masking policy. The data masking makes sure that only authorized users can see the actual data while other users will see the masked data. Since the masked data is format preserving, they can continue their work without getting access to the actual sensitive data. For example, the application developers can use masked data to develop the application whereas when the application is actually deployed, it will show actual data to the authorized user. Similarly, a security administrator may chose to mask credit card number when it is displayed to a service agent.

4.2. Row Level Filtering

The data authorization is typically required at column level as well as at row level. For example, in an organization which is geographically distributed in many locations, the security administrator may want to give access of a data from a specific location to the specific user. In other example, a hospital data security administrator may want to allow doctors to see only his or her patients. Using Ranger, such row level access control can be specified and implemented.

5. HADOOP DISTRIBUTION SUPPORT

Ranger can be deployed on top of Apache Hadoop. [4] provides detailed steps of building and deploying Ranger on top of Apache Hadoop.

Hortonwork Distribution of Hadoop (HDP) supports Ranger deployment using Ambari. [5] provides installation, deployment and configuration steps for Ranger as part of HDP deployment.

Cloudera Hadoop Distribution (CDH) does not support Ranger. According to [6], Ranger is not recommended on CDH and instead Apache Sentry should be used as central security and audit tool on top of CDH.

6. USE CASES

Apache Ranger provides centralized security framework which can be useful in many use cases as explained below.

6.1. Data Lake

[7] explains that storing many types of data in the same repository is one of the most important feature of Data Lake. With multiple datasets, the ownership, security and access control of the data becomes primary concern. Using Apache Ranger, the security administrator can define fine grain control on the data access.

6.2. Multi-tenant Deployment of Hadoop

Hadoop provides ability to store and process data from multiple tenants. The security framework provided by Apache Ranger can be utilized to protect the data and resources from un-authorized access.

7. APACHE RANGER AND APACHE SENTRY

According to [8], Apache Sentry and Apache Ranger have many features in common. Apache Sentry ([9]) provides role based authorization to data and metadata stored in Hadoop.

8. EDUCATIONAL MATERIAL

[10] provides tutorial on topics like A)Security resources B)Auditing C)Securing HDFS, Hive and HBase with Knox and Ranger D) Using Apache Atlas' Tag based policies with Ranger. [11] provides step by step guidance on getting latest code base of Apache Ranger, building and deploying it.

9. LICENSING

Apache Ranger is available under Apache 2.0 License.

10. CONCLUSION

Apache Ranger is useful to Hadoop Security Administrators since it enables the granular authorization and access control. It also provides central security framework to different data storage and access mechanism like Hive, HBase and Storm. Apache Ranger also provides audit mechanism. With Apache Ranger, the security can be enhanced for complex Hadoop use cases like Data Lake.

ACKNOWLEDGEMENTS

The authors thank Prof. Gregor von Laszewski for his technical guidance.

REFERENCES

[1] Hortonworks, “Apache ranger - overview,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: https://hortonworks.com/apache/ranger/#section_2

[2] A. S. Foundation, “Apache ranger - frequenty asked questions,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: http://ranger.apache.org/faq.html#How_does_it_work_over_Hadoop_and_related_components

[3] S. Mahmood and S. Venkat, “For your eyes only: Dynamic column masking & row-level filtering in hdp2.5,” Web Page, Sep. 2016, online; accessed 9-Mar-2017. [Online]. Available: https://hortonworks.com/blog/eyes-dynamic-column-masking-row-level-filtering-hdp2-5/

[4] A. S. Foundation, “Apache ranger 0.5.0 installation,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: https://cwiki.apache.org/confluence/display/RANGER/Apache+Ranger+0.5.0+Installation

[5] Horto, “Installing apache rang,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.3.6/bk_installing_manually_book/content/ch_installing_ranger_chapter.html

[6] Cloudera, “Configuring authorization,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: https://www.cloudera.com/documentation/enterprise/5-6-x/topics/sg_authorization.html

[7] Teradata and Hortonworks, “Putting the data lake to work - a guide to best practices,” Web Page, Apr. 2014, online; accessed 9-Mar-2017. [Online]. Available: https://hortonworks.com/wp-content/uploads/2014/05/TeradataHortonworks_Datalake_White-Paper_20140410.pdf

[8] S. Neumann, “5 hadoop security projects,” Web Page, Nov. 2014, online; accessed 9-Mar-2017. [Online]. Available: https://www.xplenty.com/blog/2014/11/5-hadoop-security-projects/

[9] A. S. Foundation, “Apache sentry,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: https://sentry.apache.org/

[10] Hortonworks, “Apache ranger overview,” Web, online; accessed 9-Mar-2017. [Online]. Available: https://hortonworks.com/apache/ranger/#tutorials

[11] A. S. Foundation, “Apache ranger - quick start guide,” Web Page, online; accessed 9-Mar-2017. [Online]. Available: http://ranger.apache.org/quick_start_guide.html

Amazon Kinesis

ABHISHEK GUPTA[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: abhigupt@iu.edu*

project-001, March 25, 2017

Amazon Kinesis [1] provides a software-as-a-service(SAAS) platform for application developers working on Amazon Web Services(AWS) platform. Kinesis is capable of processing streaming data at in real time. This is a key challenge application developers face when they have to process huge amounts of data in real time. It can scale up or scale down based on data needs of the system. As volume of data grows with advent IOT devices and sensors, Kinesis will play a key role in developing applications which require insights in real time with this growing volume of data.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IO-3005/report.pdf

1. INTRODUCTION

Amazon Kinesis [1] helps application developers collect and analyze streaming data in real time. The streaming data can come from variety of sources like social media, sensors, mobile devices, syslogs, logs, web server logs, network data etc. Kinesis can scale on demand as application needs change. For example during peak load situation kinesis added more workers nodes and can reduce the nodes when the application runs at low load. It also provides durability, where if one of the node goes down the data is persisted on disk and get replicated when new nodes come up. Multiple applications can consume data from one or more streams for variety of use cases for example, one application computes moving average and another application counts the number of users clicks. These applications can work in parallel and independently. Kinesis provides streaming in realtime with sub second delays between producer and consumer. Kinesis has two types of processing engines

- Kinesis streams - reads data from producers
- Kinesis firehose - pushes data to consumers

Kinesis streams can be used to process incoming data from multiple sources. Kinesis firehose is used to load streaming data into AWS like Kinesis analytics, S3, Redshift, Elasticsearch etc.

2. ARCHITECTURE

Amazon Kinesis reads data from variety of sources. The data coming into streams is in a record format. Each record is composed on a partition key, sequence number and data blob which is raw serialized byte array. The data further is partitioned into multiple shards(or workers) using the partition key.

Following are key components in streams architecture [2] :

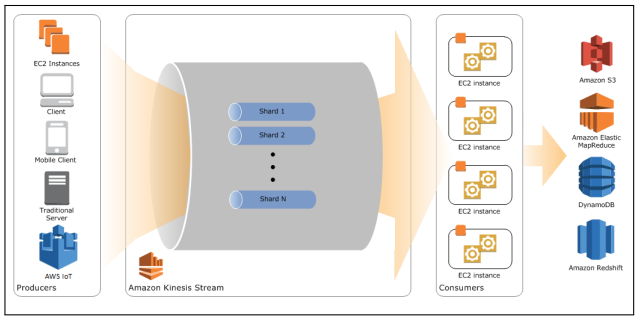


Fig. 1. Kinesis streams building blocks [2]

2.1. Data Record

Its one unit of data that flows through Kinesis stream. Data records is made up of sequence number, partition key, and blob of actual data. Size of data blob is max 1 MB. During aggregation one or more records are aggregated in to a single aggregated record. Further these aggregated records are emitted as an aggregation collection.

2.2. Producer

Producers write the data to Kinesis stream. Producer can be any system producing data. For example, ad server, social media stream, log server etc.

2.3. Consumer

Consumers subscribe to one or more streams. Consumer can be one of the applications running on AWS or hosted on EC2 instance(virtual machines).

Fig. 2. Aggregation of records

2.4. Consumer

A stream can have one or more shards. Records are processed by each shard based on the partition key. Each shard can process upto 2MB/s data for reads and upto 1MB/s for writes. Total capacity of a stream is sum of capacities of its shards.

2.5. Partition Keys

Partition key is 256 bytes long. A MD5 hash function is used to map partition keys to 128 bit integer value which is further used to map to appropriate shard.

2.6. Sequence Number

Sequence number is assigned to a record when a record get written to the stream.

2.7. Amazon Kinesis Client Library

Amazon Kinesis Client Library is bundled into your application built on AWS. It makes sure that for each record there is a shard available to process that record. Client library uses dynamo db to store control data records being processed by shards.

2.8. Application Name

Name of application is stored in the control table in dynamodb where kinesis streams will write the data to. This name is unique.

3. CREATING STREAMS

You can create a stream using following ways:

- kinesis console
- Streams API
- AWS CLI

Before creating stream you should determine initial size of the stream [3] and number of shards required to create your stream. Number of shards can be calculated using the following formulae

$$number_of_shards = max(incomingWriteBandwidthInKB/1000, outgoingReadBandwidthInKB/2000)$$

Here, the attributes used in the calculation are self explanatory.

Producer for streams writes data records into Kinesis streams. This data is available for 24 hours within streams. The default retention interval can be changed. To write records to stream, you must specify partition key, name of stream and data blob. Consumer on the other hand read data from streams using shard iterator. Shard iterator provides consumer a position on streams from where the consumer can start reading the data.

4. STREAM LIMITS

4.1. Shard

By default there can 25 shards in a region except US east, EU and US west have limit of 50 shards. Each shard can support up to 5 transactions per second for reads and at maximum data rate of 2 MB per second. Each shard can support 1000 records per second for writes and maximum data rate of 1MB per second.

4.2. Data retention

By default the data is available for 24 hours which can be configured up to 168 hours with 1 hour increments.

4.3. Data Blob

Maximum size of data blob is 1MB before base64 encoding.

5. MANAGEMENT

Kinesis provides all management using:

- AWS console
- Java SDK [4]

5.1. Java SDK

AWS provides a java SDK. Java SDK [4] can be used to complete all workflows on stream. Workflow like create, listing, retrieving shards from stream, deleting stream, resharding stream and changing data retention period. SDK provide rich documentation and developer blogs to support development on streams.

5.2. AWS console

AWS provides a web console to manage all AWS services including kinesis. Using console web user interface user can perform all operations to manage stream.

6. MONITORING

AWS provides several ways to monitor streams. These are:

- CloudWatch metrics
- Kinesis Agent
- API logging
- Client library
- Producer Library

CloudWatch metrics allows you can monitor the data and usage at shard level. It can collect metrics like: latency, coming bytes, incoming records, success count etc.

7. LICENSING

Kinesis is software as a service(SAAS) from Amazon AWS infrastructure. Hence it can only run as a service within AWS. It comes with pay as you go pricing.

8. USE CASES

Kinesis streams [1] and firehose can be useful in variety of use cases

- log data processing
- log mining
- realtime metrics and reporting
- realtime analytics
- complex stream processing

Kinesis solves variety of these business problems by doing a real time analysis and aggregation. This aggregated data can further be stored or available to query. Since it runs on amazon, it becomes easy for users to integrate and use other AWS components.

9. CONCLUSION

Kinesis can process huge amounts of data in realtime. Application developers can then focus on business logic. Kinesis [5] can help build realtime dashboards, capture anomalies, generate alerts, provide recommendations which can help take business and operation decisions in real time. It can also send data to other AWS services like S3, dynamodb, redshift etc. You can scale up or scale down as application demand increases or decreases and only pay based on your usage. Only downside of Kinesis it that it cannot run on a private or hybrid cloud, rather can only run on AWS public cloud or Amazon VPC(Virtual Private Cloud). Customers who want to use Kinesis but don't want to be on Amazon platform cannot use it. A comparable alternative is open source Apache Kafka [6] . However, Kafka lacks in high availability and monitoring in case of cloud deployments.

ACKNOWLEDGEMENTS

Special thanks to Professor Gregor von Laszewski, Dimitar Nikolov and all associate instructors for all help and guidance related to latex and bibtex, scripts for building the project, quick and timely resolution to any technical issues faced. The paper is written during the course I524: Big Data and Open Source Software Projects, Spring 2017 at Indiana University Bloomington.

REFERENCES

[1] "Kinesis - real-time streaming data in the aws cloud," Web Page, accessed: 2017-01-17. [Online]. Available: https://aws.amazon.com/kinesis/

[2] "Amazon Kinesis streams key concepts," Web Page, accessed: 2017-03-15. [Online]. Available: http://docs.aws.amazon.com/streams/latest/dev/key-concepts.html

[3] "Kinesis - real-time streaming data in the aws cloud," Web Page, accessed: 2017-03-15. [Online]. Available: http://docs.aws.amazon.com/streams/latest/dev/amazon-kinesis-streams.html

[4] "Kinesis - aws sdk for java," Web Page, accessed: 2017-03-15. [Online]. Available: https://aws.amazon.com/sdk-for-java/

[5] J. Varia and S. Mathew, "Overview of amazon web services," *Amazon Web Services*, pp. 1–22, Jan. 2014. [Online]. Available: http://w.emacromall.com/techpapers/Overview%20of%20Amazon%20Web%20Services.pdf

[6] P. Deyhim, "Kafka vs. kinesis," Datapipe, Inc., techreport, 2016, last accessed: 2017-03-17. [Online]. Available: http://go.datapipe.com/whitepaper-kafka-vs-kinesis.pdf

Robot Operating System (ROS): A Useful Overview

MATTHEW LAWSON[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: laszewski@gmail.com*

paper2, March 25, 2017

The Open Source Robotics Foundation (OSRF) oversees the maintenance and development of the Robot Operating System (ROS). ROS provides an open-source, extensible framework upon which roboticists can build simple or highly complex operating programs for robots. Features to highlight include: a) ROS' well-developed, standardized intra-robot communication system; b) its sufficiently-large set of programming tools; c) its C++ and Python APIs; and, d) its extensive library of third-party packages to address a large proportion of roboticists software needs. The OSRF distributes ROS under the BSD-3 license.

Keywords: Cloud, I524, robot, ros, ROS

https://github.com/eunosm3/classes/blob/master/docs/source/format/report/report.pdf

1. ROBOT OPERATING SYSTEM (ROS)

1. Introduction The Open Source Robotics Foundation's middleware product *Robot Operating System*, or ROS, provides a framework for writing operating systems for robots. ROS offers "a collection of tools, libraries, and conventions [meant to] simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms" [**?**]. The Open Source Robotics Foundation, hereinafter OSRF or the Foundation, attempts to meet the aforementioned objective by implementing ROS as a modular system. That is, ROS offers a core set of features, such as inter-process communication, that work with or without pre-existing, self-contained components for other tasks.

2. Architecture The OSRF designed ROS as a distributed, modular system. The OSRF maintains a subset of essential features for ROS, i.e., the core functions upon which higher-level packages build, to provide an extensible platform for other roboticists. The Foundation also coordinates the maintenance and distribution of a vast array of ROS add-ons, referred to as modules. Figure 1 illustrates the ROS universe in three parts: a) the plumbing, ROS' communications infrastructure; b) the tools, such as ROS' visualization capabilities or its hardware drivers; and c) ROS' ecosystem, which represents ROS' core developers and maintainers, its contributors and its user base.

The modules or packages, which are analogous to packages in Linux repositories or libraries in other software distributions such as *R*, provide solutions for numerous robot-related challenges. General categories include a) drivers, such as sensor and actuator interfaces; b) platforms, for steering and image processing, etc.; c) algorithms, for task planning and obstacle avoidance; and, d) user interfaces, such as tele-operation and sensor data display. [**?**]

Communications Infrastructure || General OSRF maintains three distinct communication methods for ROS: a) *message passing*; b) *services*; and, c) *actions*. Each method utilizes ROS' standard communication type, the *message* [**?**]. Messages, in turn, adhere to ROS' *interface description language*, or IDL. The IDL dictates that messages should be in the form of a data structure comprised of typed fields [**?**]. Finally, *.msg* files store the structure of messages published by various nodes so that ROS' internal systems can generate source code automatically.

Communications Infrastructure || Message Passing ROS implements a publish-subscribe anonymous message passing system for inter-process communication, hereinafter pubsub, as its most-basic solution for roboticists. A pubsub system consists of two complementary pieces: a) a device, node or process, hereinafter node, publishing messages, i.e., information, to a *topic*; and b) another node *listening to* and ingesting the information from the associated topic. Pubsub's method of operation analogizes to terrestrial radio. In the analogy, the radio station represents the publishing node, the radio receiver maps to the subscribing node and the frequency on which one transmits and the other receives represents the topic.

The OSRF touts the pubsub communications paradigm as the ideal method primarily due to its anonymity and its requirement to communicate using its message format. With respect to the first point, the nodes involved in bilateral or multilateral conversations need only know the topic on which to publish or subscribe in order to communicate. As a result, nodes can be replaced, substituted or upgraded without changing a single line of code or reconfiguring the software in any manner. The subscriber node can even be deleted entirely without affecting

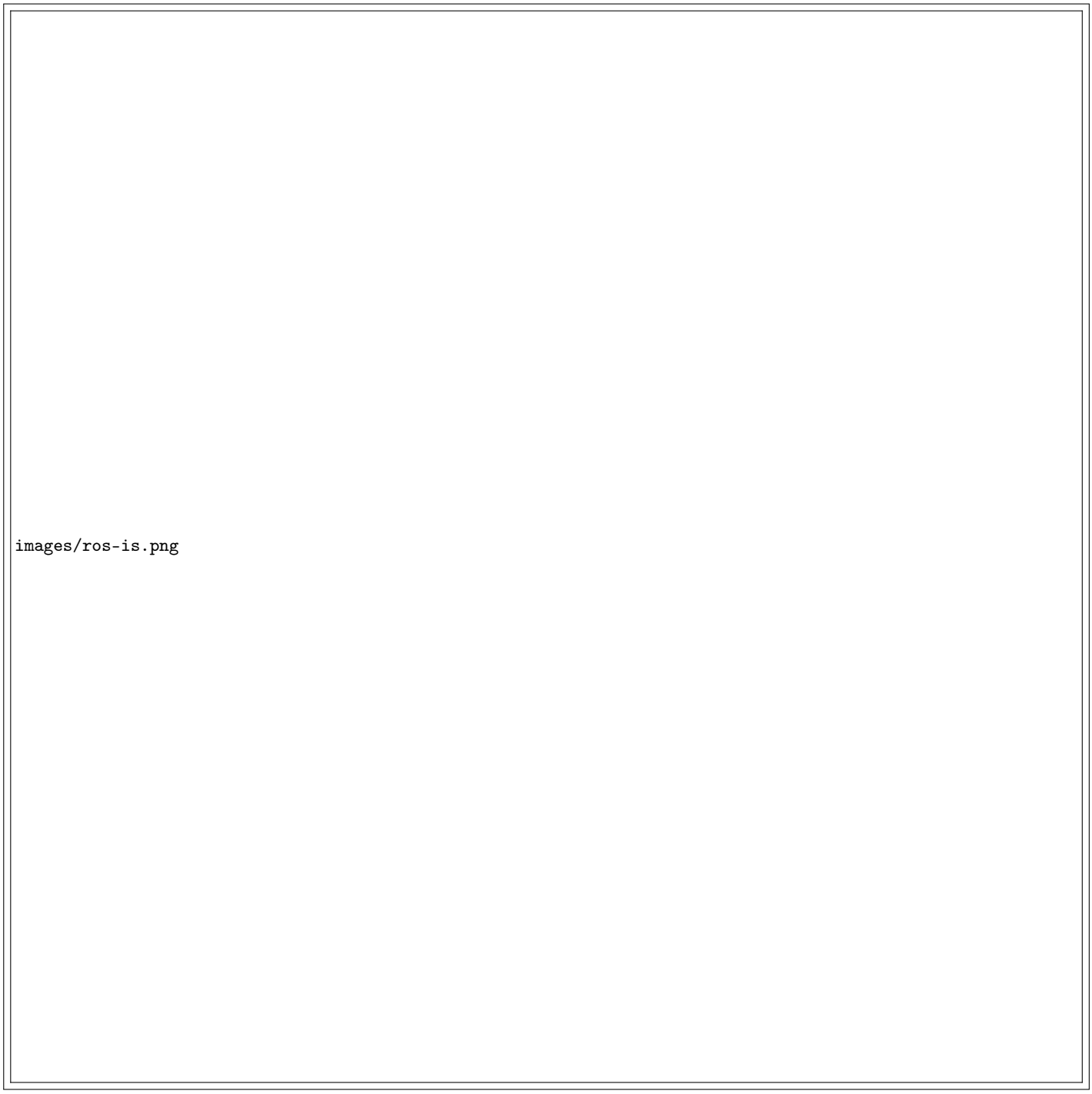


Fig. 1. A Conceptualization of What ROS, the *R*obot *O*perating *S*ystem, Offers to Roboticists [**?**]

any aspect of the robot except those nodes that depend on the deleted node.

In addition, ROS' pubsub requires well-defined interfaces between nodes in order to succeed. For instance, if a node publishes a message without a crucial piece information a subscribing node requires or in an unexpected format, the message would be useless. Alternatively, it would be pointless for an audio processing node to subscribe to a node publishing lidar data. Therefore, a message's structure must be well-defined and available for reference as needed in order to ensure compatibility between publisher and subscriber nodes. As a result, ROS has a modular communication system. That is, a subscriber node may use all or only parts of a publishing node's message. Further, the subscribing node can combine the data with information from another node before publishing the combined information to a different topic altogether for a third node's use. At the same time, a fourth and fifth node could subscribe to the original topic for each node's respective purpose.

Finally, ROS' pubsub can natively replay messages by saving them as files. Since a subscriber node processes messages received irrespective of the message's source, publishing a saved message from a subscriber node at a later time works just as well as an actual topic feed. One use of asynchronous messaging: postmortem analysis and debugging.

Communications Infrastructure || Services ROS also provides a synchronous, real-time communication tool under the moniker *services* [**?**]. Services allow a subscribing node to request information from a publishing node instead of passively receiving whatever the publishing node broadcasts whenever it broadcasts it. A service consists of two messages, the request and the reply. It otherwise mirrors ROS' message passing function. Finally, users can establish a continuous connection between nodes at the expense of service provider flexibility.

Communications Infrastructure || Actions ROS *actions* offer a more-advanced communication paradigm than either message passing or services [**?**]. Actions, which use the basic message structure from message passing, allow roboticists to create a request to accomplish some task, receive progress reports about the task completion process, receive task completion notifications and / or cancel the task request. For example, the roboticist may create a task, or equivalently, initiate an action, for the robot to conduct a laser scan of the area. The request would include the scan parameters, such as minimum scan angle, maximum scan angle and scan speed. During the process, the node conducting the scan will regularly report back its progress, perhaps as a value representing the percent of the scan completed, before returning the results of the scan, which should be a point cloud.

Tools || Standard Messages ROS' extensive use in the robotics realm has allowed it to create message standards for various robot components [**?**]. Standard message definitions exist "for geometric concepts like poses, transforms, and vectors; for sensors like cameras, IMUs and lasers; and for navigation data like odometry, paths, and maps; among many others." These standards facilitate interoperability amongst robot components as well as easing development efforts by roboticists.

Tools || Robot Geometry Library Robots with independently movable components, such as appendages (with joints) or movable sensors, must be able to coordinate such movements in order to be usable. Maintaining an accurate record of where a movable component is in relation to the rest of the robot presents a significant challenge in robotics [**?**].

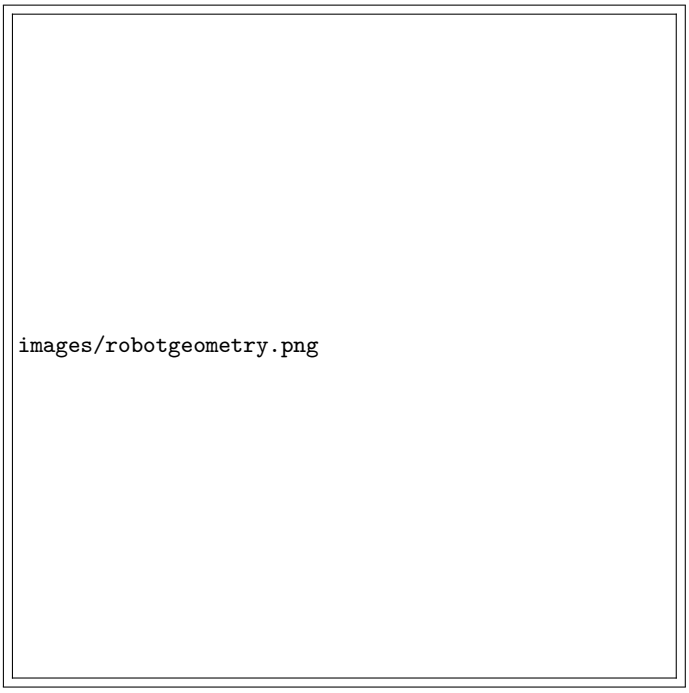


Fig. 2. A Simulated Robot with Many Coordinate Frames [**?**]

ROS addresses this issue with its *transform* library. The *tf* library tracks components of a robot using three-dimensional coordinate frames [**?**]. It records the relationship between coordinate frame positional values at sequential points in time in a tree structure. tf's built-in functions allow the roboticist to transform a particular coordinate frame's values to same basis as a different coordinate frame's values. As a result, the user, or the user's program, can always calculate any coordinate frame's relative position to any or all of the other coordinate frame positions at any point in time. Although the first-generation library, *tf*, has been deprecated in favor of the second-generation one, *tf2*, the Foundation and ROS users still refer to the library as tf.

Tools || Robot Description Language ROS describes robots in a machine-readable format using its *Unified Robot Description Format*, or URDF [**?**]. The file delineates the physical properties of the robot in XML format. URDF files enable use of the tf library, useful visualizations of the robot and the use of the robot in simulations.

Tools || Diagnostics ROS' diagnostics meta-package, i.e., a package of related packages, "contains tools for collecting, publishing, analyzing and viewing diagnostics data [**?**]." ROS' diagnostics take advantage of the aforementioned message system to allow nodes to publish diagnostic information to the standard diagnostic topic. The nodes use the diagnostic_updater and self_test packages to publish diagnostic information, while users can access the information using the rqt_robot_monitor package. ROS does not require nodes to include certain information in their respective publications, but diagnostic publications generally provide some standard, basic information. That information

may include serial numbers, software versions, unique incident IDs, etc.

Tools | | Command Line Interfaces (CLI) ROS provides at least 45 command line tools to the roboticist [**?**]. Therefore, ROS can be setup and run entirely from the command line. However, the GUI interfaces remain more popular among the user-base. Examples of ROS CLI tools include: a) *rosmsg*, which allows the user to examine messages, including the data structure of .msg files [**?**]; b) *rosbag*, a tool to perform various operations on .bag files, i.e., saved node publications; and, c) *rosbash*, which extends bash, a Linux shell program, with ROS-related commands.

Tools | | Graphical User Interfaces (GUI) OSRF includes two commonly-used GUIs, *rviz* and *rqt* [**?**], in the core ROS distribution. rviz creates 3D visualizations of the robot, as well as the sensors and sensor data specified by the user. This component renders the robot in 3D based on a user-supplied URDF document. If the end-user wants or needs a different GUI, s/he can use rqt, a Qt-based GUI development framework. It offers plug-ins for items such as: a) viewing layouts, like tabbed or split-screens; b) network graphing capabilities to visualize the robot's nodes; c) charting capabilities for numeric values; d) data logging displays; and, e) topic (communication) monitoring.

Ecosystem ROS benefits from a wide-ranging network of interested parties, including core developers, package contributors, hobbyists, researchers and for-profit ventures. Although quantifiable use metrics for ROS remain scarce, ROS does have more than 3,000 software packages available from its community of users [**?**], ranging from proof-of-concept algorithms to industrial-quality software drivers. Corporate users include large organizations such as Bosch (Robert Bosch GmbH) and BMW AG, as well as smaller companies such as ClearPath Robotics, Inc. and Stanley Innovation. University users include the Georgia Institute of Technology and the University of Arizona, among others [**?**].

3. API ROS supports robust application program interfaces, APIs, through libraries for C++ and Python. It provides more-limited, and experimental, support for nodejs, Haskell and Mono / .NET programming languages, among others. The latter library opens up use with C# and Iron Python [**?**].

4. Licensing The OSRF distributes the core of ROS under the standard, three-clause BSD license, hereinafter BSD-3 license. The BSD-3 license belongs to a broader class of copyright licenses referred to as *permissive licenses* because it imposes zero restrictions on the software's redistribution as long as the redistribution maintains the license's copyright notices and warranty disclaimers [**?**].

Other names for BSD-3 include: a) BSD-new; b) New BSD; c) revised BSD; d) The BSD License, the official name used by the Open Source Initiative; and, e) Modified BSD License, used by the Free Software Foundation.

Although the OSRF distributes the main ROS elements under the BSD-3 license, it does not require package contributors or end-users to adopt the same license. As a result, full-fledged ROS programs may include other types of *Free and Open-Source Software* [**?**], or FOSS, licenses. In addition, programs may depend on proprietary or unpublished drivers unavailable to the broader community.

5. Use Cases ROS' end-markets, its use cases, include manipulator robots, i.e., robotic arms with grasping units; mobile robots, such as autonomous, mobile platforms; autonomous cars; social robots; humanoid robots, unmanned / autonomous vehicles; and an assortment of other robots [**?**].

5.1. Use Cases for Big Data Fetch Robotics, Inc. offers its *Automated Data Collection Platform* robot to the market so corporations can "[g]ather environmental data more frequently and more accurately for [its] Internet of Things...and Big Data Applications. [**?**]" Fetch's system automatically collects data, such as RFID tracking (inventory management) or in-store shelf surveys. The latter service began in January 2017 when Fetch partnered with Trax Image Recognition, which makes image recognition software [**?**].

6. Educational material Those interested in learning more about OSRF's ROS should visit ROS' homepage, www.ros.org or its wiki page, wiki.ros.org In addition, ClearPath Robotics maintains a useful set of tutorials at https://goo.gl/hRmM3k. Finally, several books dedicated to programming with ROS, such as *Programming Robots with ROS: A Practical Introduction to the Robot Operating System* by Quibley, Gerkey and Smart can be purchased at retail.

7. Conclusion ROS offers a number of attractive features to its users, including a well-developed and standardized intra-robot communication system, modular design, vetted third-party additions and legitimacy via real-world applications. Although its status as open source software precludes direct support from its parent organization, OSRF, for-profit organizations and the software's active community of users provide reassurances to any roboticist worried about encountering a seemingly-insurmountable problem.

Acknowledgment I would like to thank my employer, Indiana Farm Bureau, for its support of my continuing education.

AUTHOR BIOGRAPHIES

Matthew Lawson received his BSBA, Finance in 1999 from the University of Tennessee, Knoxville. His research interests include data analysis, visualization and behavioral finance.

A. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Matthew Lawson. Matthew researched and wrote all of the material for this paper.

Apache Crunch

SCOTT MCCLARY[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: scmcclar@indiana.edu*

paper-002, March 26, 2017

Apache Crunch is a Java API that simplifies the process of developing MapReduce pipelines. This library is built on top of Apache Hadoop and Apache Spark and is therefore used in industry to develop efficient, scalable and maintainable codebases for Big Data solutions. The main benefit of Apache Crunch is that the explicit need to manage MapReduce jobs has been abstracted away. Thus, Apache Crunch alleviates much of the steep learning curve inherently within developing scalable applications that utilize a MapReduce type approach.

Keywords: Big-Data, Cloud, Hadoop, MapReduce

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IO-3011/report.pdf

1. INTRODUCTION

Apache Crunch is an open source Java API that is "built for pipelining MapReduce programs which are simple and efficient" [1]. More specifically, Crunch allows developers to write, test and run MapReduce pipelines with minimal upfront investment [1]. As explained in Section 1.2, this Java API was developed by Josh Wills at Cloudera and is based on Google's FlumeJava library [2, 3].

Apache Crunch "aims to make writing, testing, and running MapReduce pipelines easy, efficient, and even fun" [4]. This open source Java API provides a "small set of simple primitive operations and lightweight user-defined functions that can be combined to create complex, multi-stage pipelines" [4]. Apache Crunch abstracts away much of the complexity from the user by compiling "the pipeline into a sequence of MapReduce jobs and manages their execution" [4].

1.1. Advantages

As Hadoop continues to grow in popularity, the variation of data (i.e. satellite images, time series data, audio files, and seismograms) that is stored in HDFS grows as well [4]. Many of these data "formats are not a natural fit for the data schemas imposed by Pig and Hive;" therefore, "large, custom libraries of user-defined functions in Pig or Hive" or "MapReduces in Java" have to be written, which significantly "drain on developer productivity" [4]. The Crunch API provides an alternative solution, which does not inhibit developer productivity. Apache crunch integrates seamlessly into Java and therefore, allow developers full access to Java to write functions. Thus, Apache Crunch is "especially useful when processing data that does not fit naturally into relational model, such as time series, serialized object formats like protocol buffers or Avro records, and HBase rows and columns" [5].

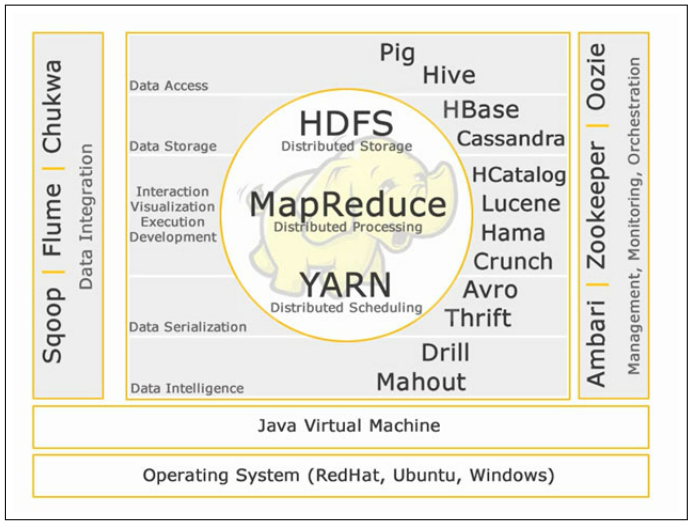


Fig. 1. The image above depicts the Apache Hadoop Ecosystem, including where Apache Crunch lies within the software stack [1].

1.2. About & History

The major contributor to the "initial source code of the Apache Crunch project [was] ... Josh Wills at Cloudera in 2011" [3]. Up until May 2012 (i.e. version 0.2.4), the Apache Crunch project was open sourced at GitHub [3]. After May 2012, "Cloudera donated the source code to Apache and the project entered

the Apache Incubator, ... [a]fter 9 months at the Incubator, ... the Apache Board of Directors established the Apache Crunch project in February 2013 as a new top level project" [3]. Since February 2013, the Apache Crunch project continues to be used, maintained and improved in an open source fashion, as explained in Section 2.1.

1.3. API

Apache Crunch is a Java API that is used "for tasks like joining and data aggregation that are tedious to implement on plain MapReduce" [5]. Section 5 explains that the Apache Software Foundation provides thorough documentation of the API and provides examples of how to explicitly leverage this API from a Java application.

1.3.1. Shell Access

For users of the Scala programming language, there is the "Scrunch API, which is built on top of the Java APIs and includes a REPL (read-eval-print loop) for creating MapReduce pipelines" [5].

2. LICENSING

The Apache Software Foundation, which includes Apache Crunch, is licensed under the Apache License, Version 2.0 [6].

2.1. Source Code

Apache Crunch leverages Git for version control, which allows the user and developer communities to contribute freely to this open source project [7].

3. ARCHITECTURE & ECOSYSTEM

Figure 1 and Figure 2 provide a complete graphical representation of the MapReduce ecosystem and explicitly indicates Apache Crunch's place within the software stack. In the simplest of terms, Apache Crunch runs on top of Hadoop MapReduce and Apache Spark [5].

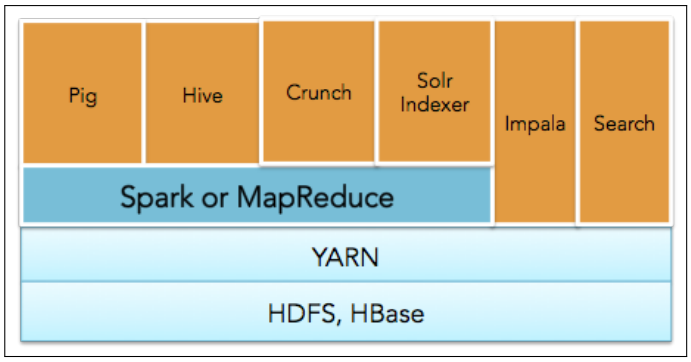


Fig. 2. The image above depicts the Apache Hadoop software stack, including Apache Crunch [8].

4. USE CASES

Apache Crunch has its applicability in the Big Data industry, as shown in Section 4.1. The widespread usage of Apache Hadoop and Apache Spark help to promote Apache Crunch in industry and academia alike.

4.1. Use Cases for Big Data

As explained in Section 3, Apache Crunch is built on top of Hadoop MapReduce and Apache Spark, which both go hand in hand in solving many complicated and challenging Big Data problems.

4.2. Cerner

Cerner, "an American supplier of health information technology (HIT) solutions, services, devices and hardware" [9], employs Apache Crunch to solve many of their Big Data problems [10]. Cerner chose to use Apache Crunch since it interestingly solves what they refer to as "a people problem" [10]. As a company, they have noticed that Apache Crunch diminishes a potential steep learning curve for new employees and/or teams to use Big Data technologies. Specifically, Apache Crunch stands above the other "options available for processing pipelines including Hive, Pig, and Cascading" since Apache Crunch allows Cerner employees to "easily ... translate how [they] ... described a problem into concepts [that they] ... can code" [10]. The diminished learning curve as a result of using Apache Crunch allows Cerner to focus their time, effort and money on performance tuning and/or algorithm adjustments rather than spending a significant amount time simply translating the problem into runnable code [10].

4.3. Spotify

Spotify, the popular "music, podcast, and video streaming service" [11], leverages Apache Crunch to process the many terabytes of data generated every day by their large user community [12]. Spotify has been using Hadoop since 2009 and have spent significant effort since then to develop tools that make it "easy for [their] developers and analysts to write data processing jobs using the MapReduce approach in Python" [12]. However, in 2013 Spotify came to the realization that this approach wasn't performing well enough so they decided to start using Java and Apache Crunch to solve their Big Data problems [12]. This transition to Apache Crunch resulted in higher performance, higher-level abstractions (e.g. filters, joins and aggregations), pluggable execution engines (e.g. MapReduce and Apache Spark) and added simple powerful testing (e.g. fast in-memory unit tests) [12]. Apache Crunch has given Spotify a "huge boost for both [their] ... developer productivity and execution performance on Hadoop" [12].

5. EDUCATIONAL MATERIAL

Apache Crunch makes the process of developing applications that leverage MapReduce and Apache Spark easier; therefore, the learning curve is much less significant in relation to developing applications that directly interact with MapReduce and Apache Spark. The Apache Software Foundation provides a lot of useful documentation. For instance, there is API documentation [13] as well as getting started information [14], a user guide [15] and even source code installation information [7]. If this is not enough, complete and extensive third-party code examples explain how to develop "hello world" applications that use Apache Crunch [16].

6. CONCLUSION

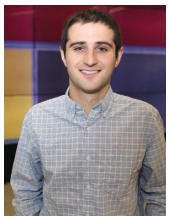
In general, Apache Crunch simplifies the process of writing and maintaining large-scale parallel codes by abstracting away the need to manage MapReduce jobs. This abstraction diminishes

the inherent learning curve in solving Big Data problems and therefore allows developers to focus their time and effort in developing the general concept of their solution rather than in the detailed process of writing their code. The aforementioned benefits of Apache Crunch are proven by its widespread use in industry (e.g. Spotify and Cerner) and in academia, shown in Section 4.

ACKNOWLEDGEMENTS

The authors would like to thank the School of Informatics and Computing for providing the Big Data Software and Projects (INFO-I524) course [17]. This paper would not have been possible without the technical support & edification from Gregor von Laszewski and his distinguished colleagues.

AUTHOR BIOGRAPHIES



Scott McClary received his BSc (Computer Science) and Minor (Mathematics) in May 2016 from Indiana University and will receive his MSc (Computer Science) in May 2017 from Indiana University. His research interests are within scientific application performance analysis on large-scale HPC systems. He will begin working as a Software Engineer with General Electric Digital in San Ramon, CA in July 2017.

WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Scott McClary. He completed all of the work for this paper including researching and testing Apache Airavata as well as composing this technology paper.

REFERENCES

[1] Edupristine, "Hadoop ecosystem and its components," Web Page, apr 2015, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/source-repository.html

[2] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. R. Robert R. Henry, R. Bradshaw, and N. Weizenbaum, "FlumeJava: Easy, Efficient Data-Parallel Pipelines," in *2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '10. Toronto, Ontario, Canada: ACM, 2010, pp. 363–375. [Online]. Available: http://doi.acm.org/10.1145/2609441.2609638

[3] The Apache Software Foundation, "Apache Crunch - About," Web Page, 2013, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/about.html

[4] J. Wills, "Introducing crunch: Easy mapreduce pipelines for apache hadoop," Blog, oct 2011, accessed: 2017-3-26. [Online]. Available: http://blog.cloudera.com/blog/2011/10/introducing-crunch/

[5] The Apache Software Foundation, "Apache Crunch Simple and Efficient MapReduce Pipelines," Web Page, 2013, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org

[6] ——, "Apache license, version 2.0," Web Page, jan 2004, accessed: 2017-3-26. [Online]. Available: http://apache.org/licenses/LICENSE-2.0.html

[7] ——, "Getting the source code," Web Page, 2013, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/source-repository.html

[8] J. Jairam Ranganathan, "Apache Spark in the Apache Hadoop Ecosystem," Blog, Sep 2014, accessed: 2017-3-26. [Online]. Available: https://vision.cloudera.com/apache-spark-in-the-apache-hadoop-ecosystem/

[9] I. Wikimedia Foundation, "Cerner - Wikipedia," Web Page, mar 2017, accessed: 2017-3-26. [Online]. Available: https://en.wikipedia.org/wiki/Cerner

[10] M. Whitacre, "Scaling people with apache crunch," Blog, may 2014, accessed: 2017-3-26. [Online]. Available: http://engineering.cerner.com/blog/scaling-people-with-apache-crunch/

[11] I. Wikimedia Foundation, "Spotify - Wikipedia," Web Page, mar 2017, accessed: 2017-3-26. [Online]. Available: https://en.wikipedia.org/wiki/Spotify

[12] J. Kestelyn, "Data processing with apache crunch at spotify," Blog, feb 2015, accessed: 2017-3-26. [Online]. Available: http://blog.cloudera.com/blog/2015/02/data-processing-with-apache-crunch-at-spotify/

[13] The Apache Software Foundation, "Apache crunch 0.15.0 api," Web Page, 2017, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/apidocs/0.15.0/

[14] ——, "Apache Crunch - Getting Started," Web Page, 2013, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/getting-started.html

[15] ——, "Apache Crunch - Apache Crunch User Guide," Web Page, 2013, accessed: 2017-3-26. [Online]. Available: https://crunch.apache.org/user-guide.html

[16] N. Asokan, "Learn Apache Crunch," Blog, Mar 2015, accessed: 2017-3-26. [Online]. Available: http://crunch-tutor.blogspot.com

[17] Gregor von Laszewski and Badi Abdul-Wahid, "Big Data Classes," Web Page, Indiana University, Jan. 2017. [Online]. Available: https://cloudmesh.github.io/classes/

Apache MRQL - MapReduce Query Language

MARK MCCOMBE[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: mmccombe@iu.edu*

project-000, March 26, 2017

Apache Map Reduce Query Language (MRQL) is a project currently in the Apache Incubator. MRQL runs on Apache Hadoop, Hama, Spark, and Flink. An overview of each dependent technology is provided along with an outline of its architecture. MRQL and MapReduce are introduced, along with a look at the MRQL language. Alternative technologies are discussed with Apache Hive and Pig highlighted. Finally, use cases for MRQL and resources for learning more more about the project are presented.

Keywords: MRQL, MapReduce, Apache, Hadoop, Hama, Spark, Flink, I524

https://github.com/cloudmesh/sp17-i524/tree/master/paper2/S17-IO-3012/report.pdf

INTRODUCTION

Apache MapReduce Query Language (MRQL) "is a query processing and optimization system for large-scale, distributed data analysis" [1]. MRQL provides a SQL like language for use on Apache Hadoop, Hama, Spark, and Flink. MapReduce Query Language allows users to perform complex data analysis using only SQL like queries, which are translated by MRQL into efficient Java code, removing the burden of writing MapReduce code directly. MRQL can evaluate queries in Map-Reduce (using Hadoop), Bulk Synchronous Parallel (using Hama), Spark, and Flink modes [1].

MRQL was created in 2011 by Leaonidas Fegaras [2] and is currently in the Apache Incubator. All projects accepted by the Apache Software Foundation (ASF) undergo an incubation period until a review indicates that the project meets the standards of other ASF projects [3]. MRQL is pronounced "miracle" [1].

ARCHITECTURE

MRQL can run on top of Apache Hadoop, Apache Hama, Apache Spark, and Apache Flink clusters. The architecture of each technology is discussed in the following sections along with the architecture of MRQL itself.

Apache Hadoop

Apache Hadoop is an open source framework written in Java that utilizes distributed storage and the MapReduce programming model for processing of big data. Hadoop utilizes commodity hardware to build fault tolerant clusters [4].

As show in Figure 1, Hadoop consists of several building blocks: the Cluster, Storage, Hadoop Distributed File System (HDFS) Federation, Yarn Infrastructure, and the MapReduce Framework. The Cluster is comprised of multiple machines, otherwise referred to as nodes. Storage can be in the HDFS or an alternative storage medium such as Amazon Web Service's Simple Storage Service (S3). HDFS federation is the framework responsible for this storage layer. YARN Infrastructure provides computational resources such as CPU and memory. The MapReduce layer is responsible for implementing MapReduce [5]. Additionally, Hadoop includes the Hadoop Common Package (not shown in Figure 1), which includes operating and file system abstractions and JAR files needed to start Hadoop [4].

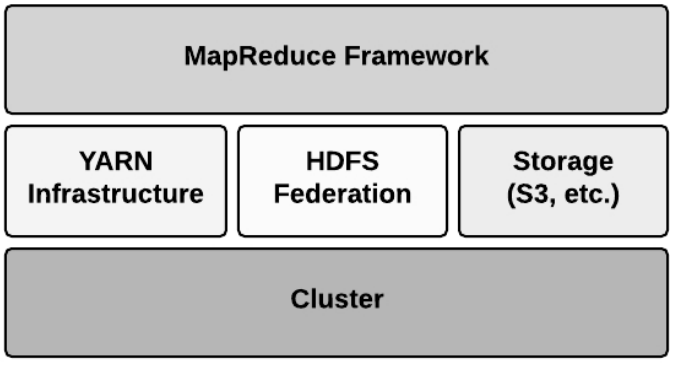


Fig. 1. Hadoop Components [5]

Figure 2 depicts a simple multi-node Hadoop cluster. The cluster contains master and slave nodes. The master node contains a DataNode, a NameNode, a Job Tracker, and a Task Tracker. The slave node functions as both a Task Tracker and a Data Node [5].

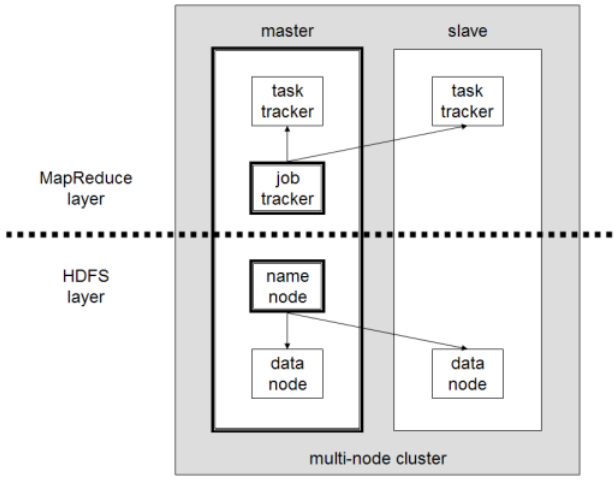


Fig. 2. Hadoop Cluster [4]

Apache Hama

Apache Hama is a top level project in the Apache Software stack developed by Edward J Yoon. Hama utilizes Bulk Synchronous Parallel (BSP) to allow massive scientific computation [6].

Figure 3 details the architecture of Apache Hama. Three key components are BSPMaster, ZooKeeper, and GroomServer. BSPMaster has several responsibilities including maintaining and scheduling jobs and communicating with the groom servers. Groom Servers are processes that perform tasks assigned by BCPMaster. Zookeeper efficiently manages synchronization of BSPPeers, instances started by groom servers [6].

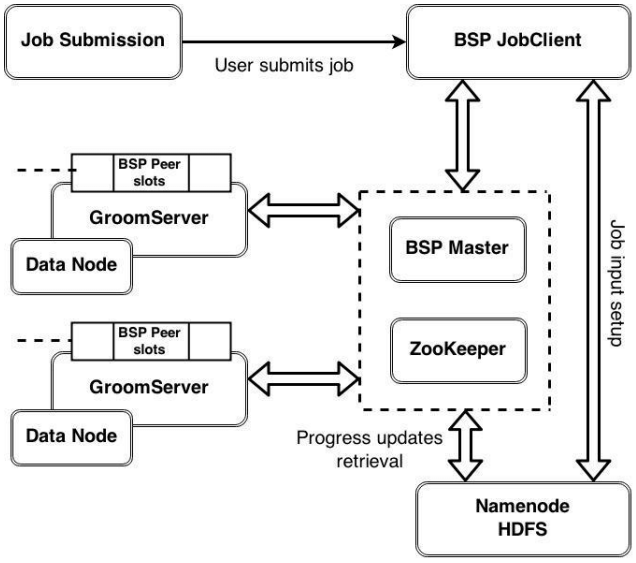


Fig. 3. Hama Architecture [7]

Apache Spark

Apache Spark is open source software, originally developed at the University of California at Berkeley and later donated to the Apache Software Foundation. Spark is a cluster computing framework providing parallelism and fault tolerance [8].

Figure 4 shows the various components of Apache Spark. Spark Core is central to Spark's architecture and provides APIs, memory management, fault recovery, storage capabilities, and other features. On top of Spark core are several packages. Spark SQL allows the use of SQL for working with structured data. Spark Streaming provides real time streaming capabilities. MLlib and GraphX allow the use of machine learning and graph algorithms [9].

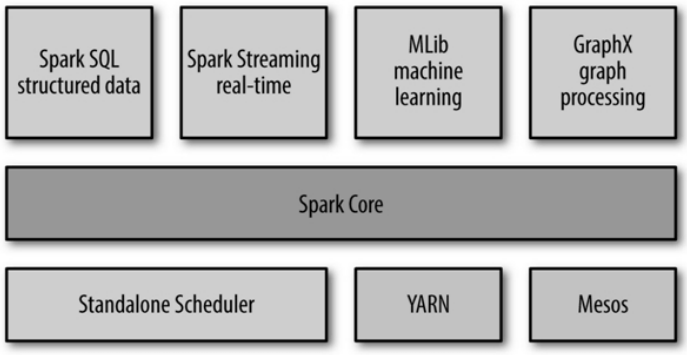


Fig. 4. Spark Architecture [9]

Apache Flink

Apache Flink is open source software developed by the Apache Software Foundation. Flink features a "distributed streaming dataflow engine written in Java and Scala" [10]. Both batch and streaming processing programs can be executed on Flink. Programs in Flink can be written in Java, Scala, Python, and SQL which are compiled and optimized and then executed on a Flink cluster. Flink does not have an internal storage system and instead provides connectors to use with external sources like Amazon S3, Apache Kafka, HDFS, or Apache Cassandra [10]. Flink's full architechure is shown in Figure 5.

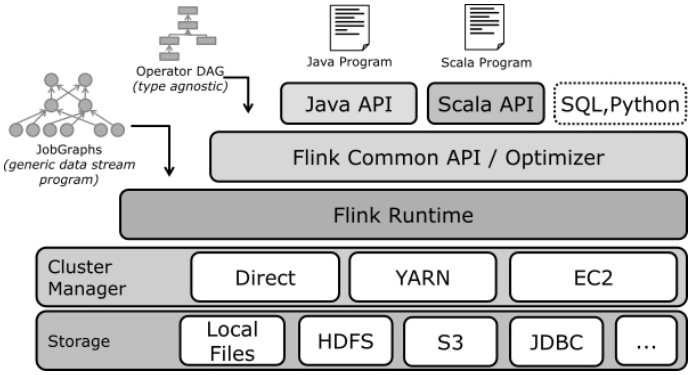


Fig. 5. Flink Architecture [11]

MRQL

MRQL itself is made up of a core module, which includes data formats and structures used by MRQL. Supporting modules, which sit on top of the core module, extend its functionalily [12]. MRQL uses a multi-step process to convert SQL statements to Jar files that are then deployed into the cluster. SQL statements are first changed into MRQL algebraic form. Next is a type interface step before the query is translated and normalized. A plan is generated, simplified, normalized, and compiled into java btye code in the final jar file [12].

MAPREDUCE

The MapReduce programing model was introduced by Google in 2004. The MapReduce model involves a map and reduce function. The map function processes key/value pairs to generate a new list of key/value pairs. The reduce function merges these new values by key [13]. MapReduce Query Language is built upon the MapReduce model.

LANGUAGE FEATURES

The MRQL language supports data types, functions, aggregation, and a SQL like syntax.

Data Model

MRQL is a typed language with several data types. Basic types (bool, short, int, long, float, double, string), tuples, records, lists, bags, user-defined types, a data type T, and persistent collections are all supported by MRQL [14].

Data Sources

MRQL can access flat files (such as CSV) and XML and JSON documents [14].

Syntax

MRQL supports a SQL like syntax. MRQL includes group by and order by clauses, nested queries, and three types of repetition [14].

```
select [ distinct ] e
from p1 in e1, ..., pn in en
[ where ec ]
[ group by p': e' [ having eh ] ]
[ order by e0 [ limit el ] ]
```

MRQL Select Query Syntax [14]

Functions and Aggregations

In addition to predefined system functions, MRQL supports user defined functions and aggregations [14].

LICENSING

MRQL is open source software licensed under the Apache 2.0 software license [15].

ALTERNATIVE TECHNOLOGIES

There are several existing MapReduce Query Languages that are alternatives to MRQL. Apache Hive (HiveQL), Apache Pig (Pig Latin), and JAQL, a JSON based query language originally developed by Google [16], are three examples. Hive and Pig, popular Apache technologies, are explored in more detail.

HiveQL - Apache Hive

Apache Hive allows the use of the use of SQL (via HiveQL) to access and modify datasets in distributed storage that integrates with Hadoop. Hive also provides a procedural language, HPL-SQL [17].

Comparing MRQL to Hive, we find several differences. Hive stores metadata in a Relational Database Management System while MRQL does not utilize metadata. MRQL allows the use of Group By on arbitrary queries. Hive does not allow the use of Group By on subqueries. MRQL runs on Hadoop, Hama, Spark, and Flink while Hive works on Hadoop, Tez, and Spark. MRQL is compatible with text, sequence, XML, and JSON file formats. Hive is compatible with test, sequence, ORC, and RCFile formats. MRQL allows iteration; Hive does not. MRQL allows streaming; Hive does not [18].

Pig Latin - Apache Pig

Apache Pig was developed at Yahoo in 2006. Like MRQL and Hive, Pig abstracts programming from Java MapReduce to a simpler format. Pig's programming language is called Pig Latin. As opposed to declarative languages where the programmer specifies what will be done like SQL, Hive, and MRQL, Pig Latin is a procedural language where the programmer specifies how the task will be accomplished [19].

Ecosystem

MRQL is part of the vast Apache ecosystem often used when working with Big Data. Other technologies in the Apache Big Data stack closely related to MRQL are Hadoop, Hama, Spark, Flink, and HDFS.

USE CASES

Due to MRQL's relatively recent development and current status in the Apache incubator, real world use cases are difficult to find. Since MRQL can be utilized with Hadoop, Hama, Spark, and Flink, it can be utilized in a wide variety of situations. MRQL can be used for complex data analysis including PageRank, matrix factorization, and k-means clustering [1].

EDUCATIONAL MATERIAL

Several excellent resources exist for learning more about MRQL. The Apache Wiki contains several research papers and presentations on MRQL by its creator Leonidas Fegaras and others [20] which provide a theoretical bases for understanding MRQL. Key resources are *Apache MRQL (incubating): Advanced Query Processing for Complex, Large-Scale Data Analysis* by Leonidas Fegaras [18], *Supporting Bulk Synchronous Parallelism in Map-Reduce Queries* by Leonidas Fegaras [21], and *An Optimization Framework for Map-Reduce Queries* by Leonidas Fegaras, Chengkai Li, and Upa Gupta [22].

The Apache Wiki also contains a *Getting Started* page [23] which describes steps such as downloading and installing MRQL, a detailed *Language Description* [14], and a listing of *System Functions* [24]. These are the best resources for hands on use of MRQL.

As MRQL is an open source technology, the source code it freely available. It is stored in github [25].

CONCLUSION

MapReduce Query Language simplifies the popular MapReduce programing model frequently utilized with Big Data. MRQL is powerful enough to express complex data analysis including PageRank, matrix factorization, and k-means clustering, yet due to its familiar SQL like syntax can be utilized by a wider variety of users without strong programming skills.

MRQL can be used with Apache Hadoop, Hama, Spark, and Flink. With Hadoop now a mainstream technology and Hama, Spark, and Flink important pieces of the Apache Big Data stack, potential applications of MRQL are wide ranging.

The declarative MRQL language is easy to use, yet powerful, featuring multiple data types, data sources, functions, aggregations, and a SQL like syntax, including order by, group by, nested queries, and repetition.

While currently still an incubator project at Apache, MRQL shows promise as a rich, easy to use language with the flexibility of working with Hadoop, Hama, Spark, and Flink. Due to these strengths, MRQL may emerge as a viable alternative to currently more popular high level MapReduce abstractions such as Hive and Pig.

REFERENCES

[1] Apache Software Foundation, "Apache incubator," Web Page, accessed 2017-01-29. [Online]. Available: http://incubator.apache.org/

[2] E. J. Yoon, "Mrql - a sql on hadoop miracle," Web Page, Jan. 2017, accessed 2017-01-29. [Online]. Available: http://www.hadoopsphere.com/2013/04/mrql-sql-on-hadoop-miracle.html

[3] Apache Software Foundation, "Apache mrql," Web Page, Apr. 2016, accessed 2017-01-29. [Online]. Available: https://mrql.incubator.apache.org/

[4] Wikipedia, "Apache hadoop," Web Page, Mar. 2017, accessed 2017-03-18. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Hadoop

[5] E. Coppa, "Hadoop architecture overview," Code Repository, accessed 2017-03-23. [Online]. Available: http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html

[6] Wikipedia, "Apache hama," Web Page, Dec. 2014, accessed 2017-03-20. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Hama

[7] K. Siddique, Y. Kim, and Z. Akhtar, "Researching apache hama: A pure bsp computing framework," vol. 2, no. 2. World Academy of Science, Engineering and Technology, 2015, p. 1857. [Online]. Available: http://waset.org/abstracts/Computer-and-Information-Engineering

[8] Wikipedia, "Apache spark," Web Page, Feb. 2017, accessed 2017-03-20. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Spark

[9] P. Pandy, "Spark programming - rise of spark," Web Page, Aug. 2015, accessed 2017-03-22. [Online]. Available: http://www.teckstory.com/hadoop-ecosystem/rise-of-spark/

[10] Wikipedia, "Apache flink," Web Page, Mar. 2017, accessed 2017-03-20. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Flink

[11] Apache Software Foundation, "General architecture and process mode," Web Page, accessed 2017-03-18. [Online]. Available: http://spark.apache.org/docs/1.3.0/cluster-overview.html

[12] A. PAUDEL, "Integration of apache mrql query language with apache storm realtime computational system," Master's thesis, The University of Texas at Arlington, Dec. 2016. [Online]. Available: https://uta-ir.tdl.org/uta-ir/bitstream/handle/10106/26371/PAUDEL-THESIS-2016.pdf?sequence=1

[13] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251254.1251264

[14] L. Fegaras, "Mrql: A mapreduce query language," Web Page, Jul. 2014, accessed 2017-03-24. [Online]. Available: https://wiki.apache.org/mrql/LanguageDescription

[15] Apache Software Foundation, "License," Code Repository, Apr. 2013, accessed 2017-03-21. [Online]. Available: https://github.com/apache/incubator-mrql/blob/master/LICENSE

[16] Wikipedia, "Jaql," Web Page, May 2016, accessed 2017-03-20. [Online]. Available: https://en.wikipedia.org/wiki/Jaql

[17] Apache Software Foundation, "Apache hive home," Web Page, Mar. 2017, accessed 2017-03-20. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Home

[18] L. Fegaras, "Apache mrql (incubating): Advanced query processing for comples, large-scale data analysis," Web Page, Apr. 2015, accessed 2017-03-14. [Online]. Available: https://github.com/apache/incubator-mrql/blob/master/LICENSE

[19] Wikipedia, "Apache pig (programming tool)," Web Page, Aug. 2016, accessed 2017-03-22. [Online]. Available: https://en.wikipedia.org/wiki/Pig_(programming_tool)

[20] L. Fegaras, "Publications," Web Page, Apr. 2015, accessed 2017-03-24. [Online]. Available: https://wiki.apache.org/mrql/Publications

[21] ——, "Supporting bulk synchronous parallelism in map-reduce queries," in *Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, ser. SCC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1068–1077. [Online]. Available: http://lambda.uta.edu/mrql-bsp.pdf

[22] L. Fegaras, C. Li, and U. Gupta, "An optimization framework for map-reduce queries," in *Proceedings of the 15th International Conference on Extending Database Technology*, ser. EDBT '12. New York, NY, USA: ACM, 2012, pp. 26–37. [Online]. Available: http://lambda.uta.edu/mrql.pdf

[23] L. Fegaras, "Getting started with mrql," Web Page, Aug. 2016, accessed 2017-03-24. [Online]. Available: https://wiki.apache.org/mrql/GettingStarted

[24] ——, "The mrql system functions," Web Page, Oct. 2016, accessed 2017-03-24. [Online]. Available: https://wiki.apache.org/mrql/SystemFunctions

[25] Apache Software Foundation, "Apache mrql," Code Repository, accessed 2017-03-25. [Online]. Available: https://git-wip-us.apache.org/repos/asf?p=incubator-mrql.git

AUTHOR BIOGRAPHY

Mark McCombe received his B.S. (Business Administration/Finance) and M.S. (Computer Information Systems) from Boston University. He is currently studying Data Science at Indiana University Bloomington.

WORK BREAKDOWN

All work on this paper was completed solely by Mark McCombe.

Apache Derby

RIBKA RUFAEL[1,*,+]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: rrufael@umail.iu.edu*
[+] *HID: S17-IO-3016*

paper2, March 26, 2017

Apache Derby, part of Apache DB subproject, is Java based relational database management system. Apache Derby database provides storage, access and secure management of data for Java based applications. Apache Derby is an open source software and it is licensed under Apache version 2.0. Apache Derby is written in Java and it runs on any certified JVM(Java Virtual Machine). JDBC driver in embedded or netwrok server frameworks allows applications to access Apache Derby Database.

Keywords: Apache Derby, relational database management system, JDBC

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IO-3016/report.pdf

1. INTRODUCTION

Apache Derby is an open source relational database management system(RDBMS). Applications interact with Apache Derby database through JDBC(Java Database Connectivity) driver. Apache Derby is built in Java programming language which makes it platform independent. Apache Derby can be embedded into Java based application or it can be setup to run in a client/server environment. Apache Derby complies with JDBC and ANSI SQL standards. Apache Derby database allows applications to create, update, read, delete and manage data [1, 2].

Apache Derby is part of the Apache DB subproject of Apache Software Foundation. It was in August 2004 that IBM submitted the Derby code to Apache Software Foundation. Then Apache Derby became part of the Apache DB project in July 2005 [2].

2. ARCHITECTURE

There are two views that describe Apache Derby's architecture. These views are Module View and Layer/Box view [3].

2.1. Module View

Modules and Monitor are components in Apache Derby database system. A collection of distinct functionality is a module. Examples of modules in Apache Derby are lock management, error logging and JDBC driver. Lock management is responsible for controlling concurrent transactions on data objects. Once Apache Derby database is up and running any informational messages or error messages are logged. This message logging is handled by error logging module. Applications use JDBC driver to interact with Apache Derby database. A number of classes are used for the implementation of each module [3].

The monitor is responsible for managing Apache Derby database. Whenever request to modules come, the monitor is responsible for selecting appropriate module implementation depending on what the module request was and from which environment the request came from [3].

2.2. Layer/Box View

JDBC, SQL, Store and Services are the four layers in Apache Derby. The Java Database Connectivity abbreviated JDBC is an API(Application Programming Interface) which allows applications to connect to Apache Derby database. JDBC interfaces that Apache Derby implements allow applications to connect to Apache Derby. Some of the interfaces are Driver, DataSource, ConnectionPoolDataSource, XADataSource and PreparedStatement. Apache Derby JDBC has implementation of java.sql and javax.sql classes [3].

The other layer which sits below JDBC is SQL layer. Compilation and execution are the two logical parts of SQL layer. SQL statement that is invoked by an application to Apache Derby Database passes through five step compilation process. First statement is parsed with a parser created by JavaCC(Java Compiler Compiler) and tree of query nodes is created, second step is binding to resolve objects , third step is optimizing to identify the access path, in fourth step Java class is created for the statement this is then cached to be used by other connections and finally class is loaded and instance of generated statement is created. During execution, execute methods on the class instance created during compilation are called and Derby result set is returned. JDBC layer is responsible in converting the Derby result set into JDBC result set for the applications [3].

Access and raw are the two parts of the store layer. Raw data storage for data in files and pages, transaction logging

and management is handled by the raw store. The access store interfaces with SQL layer and takes care of scanning of tables and indexes, indexing, sorting, locking and etc [3].

Lock management, error logging and cache management are part of the service layer. Clock based algorithm is used by cache management. It is mainly used for caching buffer, caching compiled java classes for SQL statement implementation plans and caching table descriptors [3].

2.3. Shell Access

Shell access to Apache Derby database is achieved by ij. ij is one of java utility tools that allows performing sql scripts on Derby database in embedded or network server frameworks. ij tool can be used for creating database, connecting to database, run and execute sql scripts. Commands in ij are case sensitive and semicolon is used to mark end of command. Other utility tools that come with Apache Derby are sysinfo , dblook and SignatureChecker. sysinfo is used to get version and other information about Apache Derby and the Java environment. dblook utility is used to generate Data Definition Language(DDL) for Derby database. Checks, functions, indexes, jar files, primary keys, foreign keys and schemas are the objects generated by dblook. SignatureChecker is used to check whether functions and procedures used in Derby database comply with the rules and standards [4].

2.4. API

Applications can interact with Apache Derby database through two JDBC drivers org.apache.derby.jdbc.EmbeddedDriver and org.apache.derby.jdbc.ClientDriver for embedded and network server frameworks respectively [4].

3. INSTALLATION

Java 2 Standard Edition (J2SE) 6 or higher is needed for installing Apache Derby and for running Derby the Java Runtime Environment (JRE) is needed. Apache Derby can be downloaded from the Apache DB download page [5]. Apache Derby can be installed on Windows, MAC, UNIX and Linux operating systems. After downloading the zipped installation file, the file should be extracted into directory and then DERBY_INSTALL variable should be set in the same directory as where Derby was installed. Apache Derby provides two frameworks Embedded Derby and Derby Network Server [6].

3.1. Embedded Derby

In the embedded Derby framework, Apache Derby engine and the application which accesses it run on the same JVM(Java Virtual Machine). Embedded Derby JDBC driver is used by the application to interact with Apache Derby database. To setup Embedded Derby mode, derby.jar and derbytools.jar must be included in the CLASSPATH after installation. The Derby engine and Embedded Derby JDBC driver are included in derby.jar. derbytools.jar includes ij tool(utility tool which can be used as scripting tool to interact with Derby database). In Embedded Derby framework, multiple users that are running in the same JVM can access the same database. Figure 1 shows the Embedded Derby framework [6].

3.2. Derby Network Server

Derby Network Server framework allows multiple application running in the same JVM or different JVMs to access Derby

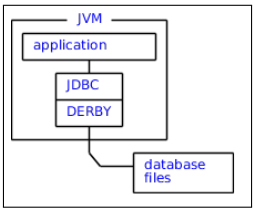


Fig. 1. Embedded Derby Framework [6].

database over the netwrok in a typical client/server architecture. In this framework application accesses Derby database through Derby Network Client JDBC driver. To setup Derby netwrok server derbynet.jar and derbytools.jar must be included in CLASSPATH on the server side. The program for Network Server and reference to the Derby engine are included derbynet.jar file. On the client side, derbyclient.jar and derbytools.jar must be included in CLASSPATH. Derby Network Client JDBC driver is included in derbyclient.jar file. Derby Network Server listens and accepts requests on port 1527 by default but this can be changed to a different port if needed. Figure 2 shows the Derby Network Server framework [6].

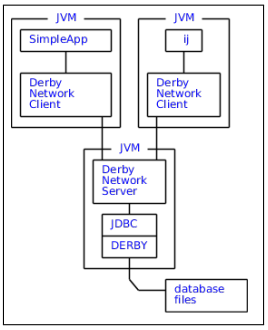


Fig. 2. Derby Network Server Framework [6].

Another variation for setting up Derby Network Server is embedded server. In embedded server, application will have both Embedded Derby JDBC driver and Network Server. The application uses Embedded Derby JDBC driver which runs on the same JVM and extends access to other applications running on different JVMs through the Network Server [6].

4. SECURITY

Apache Derby provides a number of security options. Some of these features are authentication, authorization and disk encryption. Before users are granted access to the Derby database, Derby can be setup to perform user authentication. There may be a need for certain user groups to have read only access to Derby database and some other user groups to have both read and write access to Derby database which can be achieved by Derby's user authorization feature. Data which is saved on disk can be encrypted with Derby's disk encryption feature [7].

5. USE CASES

Apache Hive, data warehouse querying and analysis software, uses Apache Derby database by default for metadata store. Apache Derby can be configured in the embedded or netwrok server mode for Hive meta data storage [8, 9].

My Money, management and analysis software for personal and business finances which is built by MTH Software, Inc, uses Apache Derby as a relational database management system [10, 11].

6. LICENSING

Apache Derby is an open source technology hence it is available for free. Apache Derby is licensed under Apache License, Version 2.0 [1].

7. EDUCATIONAL MATERIAL

Apache Derby tutorial page is one of the resources which can be used by users who are new to Apache Derby. This tutorial contains step by step information about Apache Derby installation, embedded framework configuration and network server framework configuration [6]. Derby Engine Architecture Overview page provides information about Apache Derby architecture [3]. Apache Derby documentation page has list of documentations that can be used as reference for new users, developers and administrators [12].

8. CONCLUSION

Apache Derby database offers storage,access and secure management of data for Java based applications. Apache Derby provides complete relational database management package that complies with JDBC and ANSI SQL standards. Apache Derby's small size makes it suitable for embedding it into applications which run on smaller devices with less physical memory. As the use case of Hive using Apache Derby for metadata storage indicates, Apache Derby can be incorporated into software stack for big data projects for metadata storage.

ACKNOWLEDGEMENTS

The author would like to thank Professor Gregor von Laszewski and associate instructors for their help and guidance.

REFERENCES

[1] Apache Software Foundation, "Apache Derby," Web Page, Oct. 2016, accessed: 2017-03-14. [Online]. Available: https://db.apache.org/derby/

[2] The Apache Software Foundation, "Apache Derby Project Charter," Web page, Sep. 2016, accessed: 2017-02-25. [Online]. Available: https://db.apache.org/derby/derby_charter.html

[3] Apache Software Foundation, "Derby Engine Architecture Overview," Web Page, Sep. 2016, accessed: 2017-03-14. [Online]. Available: http://db.apache.org/derby/papers/derby_arch.html#Module+View

[4] ——, "Derby Tools and Utilities Guide," Web Page, Oct. 2016, accessed: 2017-03-24. [Online]. Available: https://db.apache.org/derby/docs/10.13/tools/derbytools.pdf

[5] ——, "Apache Derby: Downloads," Web Page, Sep. 2016, accessed: 2017-03-24. [Online]. Available: http://db.apache.org/derby/derby_downloads.html

[6] ——, "Apache Derby Tutorial," Web Page, Sep. 2016, accessed: 2017-03-14. [Online]. Available: https://db.apache.org/derby/papers/DerbyTut/index.html

[7] ——, "Derby and Security," Web Page, Mar. 2017, accessed: 2017-03-20. [Online]. Available: http://db.apache.org/derby/docs/10.8/devguide/cdevcsecuree.html

[8] ——, "Apache Hive," Web Page, Mar. 2017, accessed: 2017-03-23. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Home

[9] ——, "GettingStarted-Metadata Store," Web Page, Mar. 2017, accessed: 2017-03-23. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-MetadataStore

[10] MTH Software, Inc, "MTH built products," Web Page, Mar. 2017, accessed: 2017-03-23. [Online]. Available: http://www.mthbuilt.com/products.php

[11] ——, "How to use SQL in My Money," Web Page, Mar. 2017, accessed: 2017-03-23. [Online]. Available: http://wiki.mthbuilt.com/How_to_use_SQL_in_My_Money

[12] Apache Software Foundation, "Apache Derby: Documentation," Web Page, Oct. 2016, accessed: 2017-03-24. [Online]. Available: https://db.apache.org/derby/manuals/index.html

Facebook Tao

NANDITA SATHE[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding author: nsathe@iu.edu*

paper-2, March 25, 2017

As early as 2005, Facebook started using MySQL, a relational database coupled with large distributed cache called memcache. Facebook soon realized however efficient relational database it would use, it is not sufficient to manage the enormous data challenge Facebook had. The data was a social graph. Another mismatch, which relational database or block cache had was, most of the data that would be read into cache did not belong to any relation. For example, 'If user likes that picture". In most records the answer would be 'No' or 'False'. Storing and reading this unwanted data was a burden. Meanwhile Facebook users' base was increasing daily. Ultimately Facebook came up with Facebook Tao, a distributed social graph data store.

Keywords: Facebook Tao, Graph Database, memcache

https://github.com/nsathe/sp17-i524/blob/master/paper2/S17-IO-3017/report.pdf

1. INTRODUCTION

In the paper published in USENIX annual technical conference, Facebook Inc describes TAO (The Association and Objects) as [1] a geographically distributed data store that provides timely access to the social graph for Facebook's demanding workload using a fixed set of queries. It is deployed at Facebook for many data types that fit its model. The system runs on thousands of machines, is widely distributed, and provides access to many petabytes of data. TAO represents social data items as Objects (user) and relationship between them as Associations (liked by, friend of). TAO cleanly separates the caching tiers from the persistent data store allowing each of them to be scaled independently. To any user of the system it presents a single unified API that makes the entire system appear like 1 giant graph database [2]. Key advantages of the system include [2]:

- Provides a clean separation of application/product logic from data access by providing a simple yet powerful graph API and data model to store and fetch data. This enables Facebook product engineers to move fast.
- By implementing a write-through cache TAO allows Facebook to provide a better user experience and preserve the all important read-what-you-write consistency semantics even when the architecture spans multiple geographical regions.
- By implementing a read-through write-through cache TAO also protects the underlying persistent stores better by avoiding issues like thundering herds without compromising data consistency.

2. TAO'S GOAL

Main goal of implementing Tao is efficiently scaling the data. Facebook handles approximately a billion requests per second. So obviously data store has to be scalable. More than that, scalability should be efficient otherwise scaling data across machines would be extremely costly.

Second goal is to achieve lowest possible read latency. So that if a user has commented on a post, the original post writer should be able to read it immediately. Efficiency in Scaling and low Read latency is achieved by (i) separating cache and data storage, (ii) Graph specific caching and (iii) Sub-dividing data centers [3].

Third goal is to achieve timeliness of writes. If a web server has written something and it sends a read request, it should be able to read the post. Write timeliness is achieved by (i) Write trough cache and (ii) Asynchronous replication [3].

Lastly high read availability for the same reasons mentioned above, which is achieved by using alternate data sources.

3. TAO DATA MODEL AND API

Facebook Inc. explains TAO data model and API associated with using a simple example [4]. Figure 1 depicts TAO data model.

This simple example shows a subgraph of objects and associations that is created in TAO after Alice checks in at the Golden Gate Bridge and tags Bob there, while Cathy comments on the check-in and David likes it. Every data item, such as a user, check-in, or comment, is represented by a typed object containing a dictionary of named fields. Relationships between objects, such as "liked by" or "friend of," are represented by typed edges

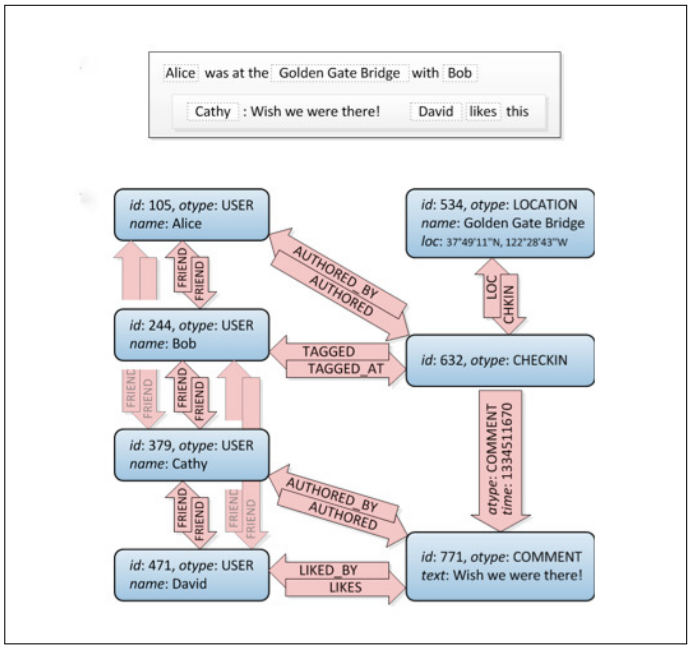


Fig. 1. TAO Data Model [4].

(associations) grouped in association lists by their origin. Multiple associations may connect the same pair of objects as long as the types of all those associations are distinct. Together objects and associations form a labeled directed multigraph.

For every association type a so-called inverse type can be specified. Whenever an edge of the direct type is created or deleted between objects with unique IDs id1 and id2, TAO will automatically create or delete an edge of the corresponding inverse type in the opposite direction (id2 to id1). The intent is to help the application programmer maintain referential integrity for relationships that are naturally mutual, like friendship, or where support for graph traversal in both directions is performance critical, as for example in “likes” and “liked by."

3.1. Objects and Associations

[1] TAO objects are typed nodes, and TAO associations are typed directed edges between objects. Objects are identified by a 64-bit integer (id) that is unique across all objects, regardless of object type (otype). Associations are identified by the source object (id1), association type (atype) and destination object (id2). At most one association of a given type can exist between any two objects. Both objects and associations may contain data as key $->$ value pairs. A per-type schema lists the possible keys, the value type, and a default value. Each association has a 32-bit time field, which plays a central role in queries.

Object: (id) -> (otype, (key -> value)*)

Assoc.: (id1, atype, id2) -> (time, (key -> value)*)

Figure 1 shows how TAO objects and associations might encode the example, with some data and times omitted for clarity. The example’s users are represented by objects, as are the checkin, the landmark, and Cathy’s comment. Associations capture the users’ friendships, authorship of the checkin and comment, and the binding between the checkin and its location and comments.

The set of operations on objects is of the fairly common create/set-fields/get/delete variety. All objects of a given type have the same set of fields. New fields can be registered for an object type at any time and existing fields can be marked deprecated by editing that type’s schema. In most cases product engineers can change the schema of their types without any operational work.

Associations are created and deleted as individual edges. If the association type has an inverse type defined, an inverse edge is created automatically. The API helps the data store exploit the creation-time locality of workload by requiring every association to have a special time attribute that is commonly used to represent the creation time of association. TAO uses the association time value to optimize the working set in cache and to improve hit rate.

There are three main classes of read operations on associations [4]:

- Point queries look up specific associations identified by their (id1, type, id2) triplets. Most often they are used to check if two objects are connected by an association or not, or to fetch data for an association.

- Range queries find outgoing associations given an (id1, type) pair. Associations are ordered by time, so these queries are commonly used to answer questions like "What are the 50 most recent comments on this piece of content?" Cursor-based iteration is provided as well.

- Count queries give the total number of outgoing associations for an (id1, type) pair. TAO optionally keeps track of counts as association lists grow and shrink, and can report them in constant time

4. TAO ARCHITECTURE

This section describes the architecture of TAO. TAO is separated into layers: two caching layers and a storage layer.

4.1. Storage Layer

The data is persisted using MySQL. The API is mapped to a small number of SQL queries. Data is divided into logical shards. By default all object types are stored in one table and association in others. Every ’object-id’ has a corresponding ’shard-id’. Objects are bounded to a single shard throughout their lifetime. An association is stored on the shard of its id1, so that every association query can be served from a single server [3].

4.2. Caching Layer

TAO’s cache implements the complete API for clients, handling all communication with databases. A region/tier is made of multiple closely located Data centers. Multiple Cache Serves make up a tier (set of databases in a region are also called a tier) that can collectively capable of answering any TAO Request. Each cache request maps to a server based on sharding. The cache is filled based on a LRU policy. Write operations on an association with an inverse may involve two shards, since the forward edge is stored on the shard for id1 and the inverse edge is on the shard for id2. Handling writes with multiple shards involve: Issuing an RPC call to the member hosting id2, which will contact the database to create the inverse association. Once the inverse write is complete, the caching server issues a write to the database for id1. TAO does not provide atomicity between the two updates. If a failure occurs the forward may exist without an inverse, these hanging associations are scheduled for repair by an asynchronous job [3].

4.3. Leaders and Followers

There are two tiers of caching clusters in each geographical region. Clients talk to the first tier, called followers. If a cache miss occurs on the follower, the follower attempts to fill its cache from a second tier, called a leader. Leaders talk directly to a MySQL cluster in that region. All TAO writes go through followers to leaders. Caches are updated as the reply to a successful write propagates back down the chain of clusters. Leaders are responsible for maintaining cache consistency within a region. They also act as secondary caches, with an option to cache objects and associations in Flash [4].

4.4. Scaling Geographically

High read workload scales with total number of follower servers. The assumption is that latency between followers and leaders is low. Followers behave identically in all regions, forwarding read misses and writes to the local region's leader tier. Leaders query the local region's database regardless of whether it is the master or slave. This means that read latency is independent of inter-region latency. Writes are forwarded by the local leader to the leader that is in the region with the master database. Read misses by followers are 25X as frequent as writes in the workload thus read misses are served locally. Facebook chooses data center locations that are clustered into only a few regions, where the intra-region latency is small (typically less than 1 millisecond). It is then sufficient to store one complete copy of the social graph per region.

Since each cache hosts multiple shards, a server may be both a master and a slave at the same time. It is preferred to locate all of the master databases in a single region. When an inverse association is mastered in a different region, TAO must traverse an extra inter-region link to forward the inverse write. TAO embeds invalidation and refill messages in the database replication stream. These messages are delivered in a region immediately after a transaction has been replicated to a slave database. Delivering such messages earlier would create cache inconsistencies, as reading from the local database would provide stale data. If a forwarded write is successful then the local leader will update its cache with the fresh value, even though the local slave database probably has not yet been updated by the asynchronous replication stream. In this case followers will receive two invalidates or refills from the write, one that is sent when the write succeeds and one that is sent when the write's transaction is replicated to the local slave database [3].

5. EDUCATIONAL MATERIAL

To get started on learning Facebook TAO, following resources can prove helpful.

- [1] - Technical paper on Facebook TAO.
- [4] - Background, Architecture and Implementation from Facebook itself.
- [5] - TAO summary in a video on USENIX website.

6. ACKNOWLEDGEMENTS

The author would like to thank Prof. Gregor von Laszewski and his associates from the School of Informatics and Computing for providing all the technical support and assistance.

REFERENCES

[1] N. Bronson *et al.*, "Tao: Facebook's distributed data store for the social graph," in *2013 USENIX Annual Technical Conference*, 2013. [Online]. Available: http://ai2-s2-pdfs.s3.amazonaws.com/39ac/2e0fc4ec63753306f99e71e0f38133e58ead.pdf

[2] V. Venkataramani, "What is the tao cache used for at facebook," Web Page, June 2013. [Online]. Available: https://www.quora.com/What-is-the-TAO-cache-used-for-at-Facebook

[3] N. Upreti, "Facebook's tao and unicorn data storage and search platforms," Slides, April 2015. [Online]. Available: https://www.slideshare.net/nitishupreti/faceboko-tao-unicorn

[4] M. Marchukov, "Tao: The power of the graph," Web Page, June 2013. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/tao-the-power-of-the-graph/10151525983993920/

[5] N. Bronson, "Tao: Facebook's distributed data store for the social graph," Slides, June 2013. [Online]. Available: https://www.usenix.org/node/174510

AWS Lambda

KARTHICK VENKATESAN[1,*,+]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: vkarthickprabu@gmail.com
[+] HID - S17-IO-3023

paper1, March 26, 2017

The rapid pace of innovation in datacenters and the software platforms within them is set to transform how we build, deploy, and manage online applications and services. Common to both hardware-based and container-based. Virtualization is the central notion of a server. Servers have long been used to back online applications, but new cloud-computing platforms foreshadow the end of the traditional backend server. Servers are notoriously difficult to configure and manage, and server startup time severely limits an application's ability to scale up and down quickly. As a result, a new model, called serverless computation, is poised to transform the construction of modern, scalable applications ability to quickly scale up and down [1]. This paper is on AWS Lambda a Serverless Computing technology.

Keywords: AWS Lambda,Serverless Computing, I524

https://github.com/karvenka/sp17-i524/tree/master/paper2/S17-IO-3023/report.pdf

1. INTRODUCTION

Serverless applications are where some amount of server-side logic are written by the application developer but unlike traditional architectures is run in stateless compute containers that are event-triggered, ephemeral, and fully managed by a 3rd party. One way to think of this is 'Functions as a service / Faas'. AWS Lambda is one of the most popular implementations of Faas [2].

AWS Lambda is a FaaS(Function as a Service) from Amazon Web Services. It runs the backend code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance and capacity provisioning [3].

In AWS Lambda one needs to pay only for the compute time consumed - there is no charge when the code is not running. AWS Lambda, can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda requires only the code to be uploaded, and Lambda takes care of everything required to run and scale the code with high availability. AWS Lambda can be setup to automatically trigger the code from other AWS services or call it directly from any web or mobile app [4].

2. FEATURES

The key features of AWS Lambda are [4]

- **No Servers to Manage**: AWS Lambda automatically runs the code without requiring to provision or manage servers. Just write the code and upload it to Lambda.
- **Continuous Scaling**: AWS Lambda automatically scales the application by running code in response to each trigger. The code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.
- **Subsecond Metering**: With AWS Lambda, the institution using the service are charged for every 100ms the code executes and the number of times the code is triggered. They don't pay anything when the code isn't running.

3. ARCHITECTURE

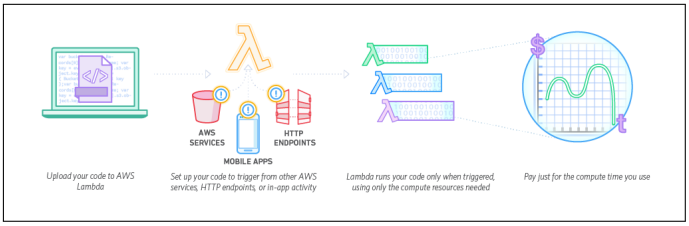


Fig. 1. AWS Lambda Architecture [4]

Lambda functions and event sources are the core components in AWS Lambda. An event source is the entity that publishes events, and a Lambda function is the custom code that processes the events. Several AWS cloud services can be preconfigured to work with AWS Lambda. The configuration is referred to as event source mapping, which maps an event source to a Lambda

function. It enables automatic invocation of Lambda function when events occur.

Each event source mapping identifies the type of events to publish and the Lambda function to invoke when events occur. The specific Lambda function then receives the event information as a parameter, and as shown in Figure 1 the Lambda function code can then process the event.

The event sources can be any of the following:

AWS services – These are the supported AWS services that can be preconfigured to work with AWS Lambda. These services can be grouped as regular AWS services or stream-based services. Amazon Kinesis Streams [5] and Amazon DynamoDB Streams [6] are stream-based event sources, all others AWS services do not use stream-based event sources.

Custom applications – Custom applications built can also publish events and invoke a Lambda function.

4. SCALABILITY

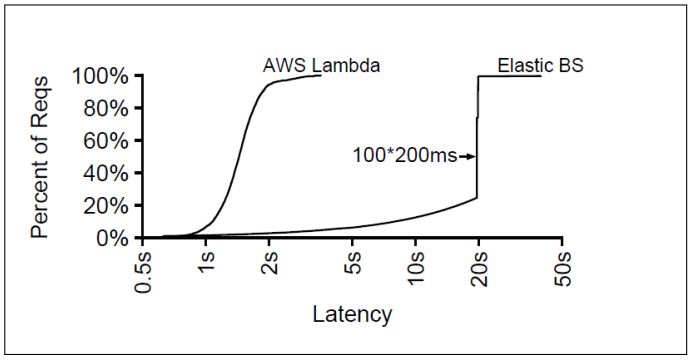


Fig. 2. Response Time. This CDF shows measured response times from a simulated load burst to an Elastic BS application and to an AWS Lambda application.
[1]

A primary advantage of the Lambda model is its ability to quickly and automatically scale the number of workers when load suddenly increases. The Graph in Figure 2 demonstrates this by comparing AWS Lambda to a container-based server platform, AWS Elastic Beanstalk [7] (hereafter Elastic BS). On both platforms, the same benchmark was run for one minute: the workload maintains 100 outstanding RPC(Remote Procedure Calls) requests and each RPC handler spins for 200ms. Figure 2 shows the result: an RPC using AWS Lambda has a median response time of only 1.6s, whereas an RPC in Elastic BS often takes 20s. Investigating the cause for this difference, it was found that while AWS Lambda was able to start 100 unique worker instances within 1.6s to serve the requests; all Elastic BS requests were served by the same instance; as a result, each request in Elastic BS had to wait behind 99 other 200ms requests. AWS Lambda also has the advantage of not requiring configuration for scaling. In contrast, Elastic BS con- figuration is complex, involving 20 different settings for scaling alone. Even though the Elastic BS was tuned to scale as fast as possible, it still failed to spin up new workers for several minutes [1].

5. DOCUMENTATION

- Detailed documentation on AWS Lambda Deployment , Configuration,Debugging and Development is available at [8].
- Use cases on reference architecture with AWS Lambda are available at [9].

6. COMPETITORS

The main competitors for AWS Lambda are

- Microsoft Azure Functions
- Google Cloud

A detailed comparison of features between AWS Lambda, Google Cloud and Microsoft Azure Functions is available in Table 1.

7. PRICING

With AWS Lambda, the users pay only for what they use. They are charged based on the number of requests for the functions and the duration the code executes.

Requests: The users are charged for the total number of requests across all their functions. Lambda counts a request each time it starts executing in response to an event notification or invokes call, including test invokes from the console. First 1 million requests per month are free $0.20 per 1 million requests thereafter ($0.0000002 per request)

Duration:Duration is calculated from the time the code begins executing until it returns or otherwise terminates, rounded up to the nearest 100ms. The price depends on the amount of memory the user allocates to the function. The users are charged $0.00001667 for every GB-second used [11].

8. USE CASE

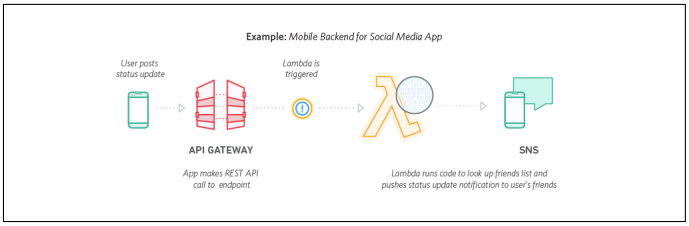


Fig. 3. Mobile Backend
[4]

Mobile Backends: Mobile Backends are an excellent use case to implement AWS Lambda.As shown in Figure 3 the back-end of the mobile application can be built using AWS Lambda and Amazon API Gateway to authenticate and process API requests. Lambda makes it easy to create rich, personalised app experiences. **Bustle.com** is a news, entertainment, lifestyle, and fashion website catering to women. Bustle also operates **Romper.com**, a website focused on motherhood.Bustle is based in Brooklyn, NY and is read by 50 million people each month. Bustle uses AWS Lambda to process high volumes of site metric data from Amazon Kinesis Streams [5] in real time. This allows the Bustle team to get data more quickly so they can understand how new site features affect usage. They can also measure user engagement, allowing better data-driven decisions. The serverless back end supports the Romper website and iOS app as well as the Bustle iOS app. With AWS Lambda, Bustle was able to eliminate the need to worry about operations. The engineering team at Bustle focus on just writing code, deploy it, and it

Table 1. AWS Lambda vs. Google Cloud Functions vs. Microsoft Azure Functions [10]

FEATURE	AWS LAMBDA	GOOGLE CLOUD	AZURE FUNCTIONS
Scalability & availability	Automatic scaling (transparently)	Automatic scaling	Metered scaling (App Service Plan) Automatic scaling (Consumption Plan)
Max # of functions	Unlimited functions	20 functions per project (alpha)	Unlimited functions
Concurrent executions	100 parallel executions (soft limit)	No limit	No limit
Max execution	300 sec (5 min)	No limit	300 sec (5 min)
Supported languages	JavaScript, Java Python, C#	Only JavaScript	C# and JavaScript (preview of F#, Python, Batch, PHP, PowerShell)
Dependencies	Deployment Packages	npm package.json	Npm, NuGet
Deployments	Only ZIP upload (to Lambda or S3)	ZIP upload, Cloud Storage or Cloud Source Repositories	Visual Studio Team Services, OneDrive, GitHub, Bitbucket, Dropbox
Environment variables	Yes	Not yet	App Settings and ConnectionStrings from App Services
Versioning	Versions and aliases	Cloud Source branch/tag	Cloud Source branch/tag
Event-driven	S3, SNS, SES, DynamoDB, Kinesis, CloudWatch, Cognito, API Gateway	Cloud Pub/Sub or Cloud Storage Object Change Notifications	Blob, EventHub, Generic WebHook Queue, Http, Timer triggers
HTTP(S) invocation	API Gateway	HTTP trigger	HTTP trigger
Orchestration	AWS Step Functions	Not yet	Azure Logic Apps
Logs management	CloudWatch	Cloud Logging	App Services monitoring
In-browser code editor	Yes	Only with Cloud Source Repositories	Functions environment, AppServices editor
Granular IAM	IAM roles	Not yet	IAM roles
Pricing	1M requests for free (Free Tier), then $0.20/1M requests	Unknown until open beta	1 million requests and 400,000 GB-s

scales infinitely, and they don't have to deal with infrastructure management. Bustle was able to achieve the same level of operational scale with half the size of team of what is normally needed to build and operate the site [12].

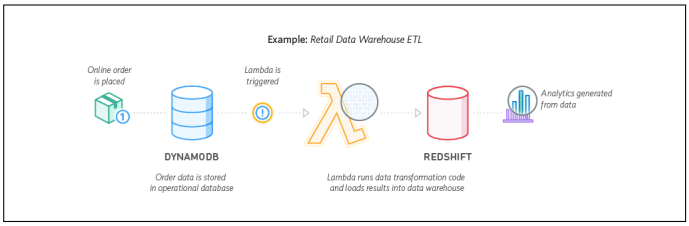


Fig. 4

[4]

Extract, Transform, Load: As shown in Figure 4 AWS Lambda can be used to perform data validation, filtering, sorting, or other transformations for every data change in a DynamoDB table and load the transformed data to another data store. **Zillow** is the leading real estate and rental marketplace dedicated to empowering consumers with data, inspiration and knowledge around the place they call home, and connecting them with the best local professionals who can help. Zillow needed to collect a subset of mobile app metrics in realtime and report it to the buisness users several times during the day.The solution was to be delivered in 3 weeks.Leveraging AWS Lambda and Amazon Kinesis Zillow was able to seamlessly scale 56 lines of code to over 16 million posts a day and achieve its goal in 2 weeks [13].

9. CONCLUSION

Serverless Computing allows to build and run applications and services without thinking about servers. At the core of serverless computing is AWS Lambda, which lets to build auto-scaling, pay-per-execution, event-driven apps quickly.

ACKNOWLEDGEMENTS

The authors thank Prof. Gregor von Laszewski for his technical guidance.

REFERENCES

[1] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in *8th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2016, Denver, CO, USA, June 20-21, 2016*. Berkeley, CA, USA: USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/hotcloud16/workshop-program/presentation/hendrickson

[2] M. Roberts, “Serverless,” Web Page, Aug. 2016, accessed 2017-03-26. [Online]. Available: https://martinfowler.com/articles/serverless.html
[3] A. Gupta, “Serverless faas with aws lambda and java,” Web Page, Jan. 2017, accessed 2017-03-26. [Online]. Available: https://blog.couchbase.com/serverless-faas-aws-lambda-java/
[4] Amazon Web Services, Inc, “AWS Lambda,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/lambda/
[5] ——, “AWS Kinesis Streams,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/kinesis/streams/
[6] ——, “AWS Dynamo DB Streams,” Web Page, accessed 2017-03-26. [Online]. Available: http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Types_Amazon_DynamoDB_Streams.html
[7] ——, “AWS Elastic Beanstalk,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/elasticbeanstalk/
[8] ——, “AWS Lambda Doc,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/documentation/lambda/
[9] ——, “AWS Lambda Use Case,” Web Page, accessed 2017-03-26. [Online]. Available: http://docs.aws.amazon.com/lambda/latest/dg/use-cases.html
[10] M. Parenzan, “Microsoft azure functions vs. google cloud functions vs. aws lambda,” Web Page, Feb. 2017, accessed 2017-03-26. [Online]. Available: http://cloudacademy.com/blog/microsoft-azure-functions-vs-google-cloud-functions-fight-for-serverless-cloud-domination-continues/
[11] Amazon Web Services, Inc, “AWS Lambda Pricing,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/lambda/pricing/
[12] ——, “Bustle Case Study,” Web Page, accessed 2017-03-26. [Online]. Available: https://aws.amazon.com/solutions/case-studies/bustle/
[13] ——, “AWS re:Invent 2015 | (BDT307) Zero Infrastructure, Real-Time Data Collection, and Analytics,” https://www.youtube.com/watch?v=ygHGPnAd0Uo, Youtube, Oct. 2015, accessed 2017-03-26.

Hyper-V

ANURAG KUMAR JAIN[1] AND GREGOR VON LASZEWSKI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: laszewski@gmail.com

project-000, March 26, 2017

A hypervisor or virtual machine monitor (VMM) is computer software, firmware, or hardware, that creates and runs virtual machines. Microsoft Hyper-V Server is the hypervisor-based server virtualization product that allows users to consolidate workloads onto a single physical server [1]. Hyper-V has advantages of being scalable, secure and flexible.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/tree/master/paper2/report.pdf

INTRODUCTION

Cloud computing is a booming field and with that, the need for virtualization is also growing. Microsoft has long back stepped into the field of virtualization and is improving in the sector. It brought Microsoft Hyper-V, also codenamed as Viridian and formerly as Windows Server Virtualization to compete with VMware vSphere [2]. It is a native hypervisor which can be used to create virtual machines on x86-64 systems running Windows.

With the release of Windows 8, Hyper-V overtook Windows Virtual-PC as the hardware virtualization component of the client editions of Windows NT. Hyper-V is also available on the Xbox One, in which it would launch both Xbox OS and Windows 10 [2]. Hyper-V supports Windows XP, Vista, Windows 7, Windows 8-8.1, Windows 10, Windows Server 2003-2016, CentOs 5.5-7.0, Red Hat Enterprise Linux 5.5-7.0, Ubuntu 12.04-14.04 among others [3].

ARCHITECTURE

Hyper-V maintains isolation of virtual machines in terms of a partition [2]. A partition is a logical unit of isolation in which each guest OS executes. A Hyper-V instance needs to have at least one parent partition, running a supported version of Windows Server (2008 and later). The virtualization stack runs in the parent partition and has direct access to the hardware devices. The child partitions, which host the guest operating systems, are created on parent partitions. A child partition is created by parent partition using the hypercall API, which is the application programming interface exposed by Hyper-V [4].

A child partition does not have direct access to the physical processor. A child partition doesn't even handle its real interrupts. It has a virtual view of the processor and runs in a guest virtual address. Depending on virtual machine configuration, Hyper-V may allow access to a subset of the processors to each partition. The hypervisor handles the interrupts to the processor, and redirects them to the respective partition. Hyper-V can hardware accelerate the address translation of guest virtual address-spaces by using second level address translation provided by the CPU [1].

Direct access to hardware resources is not allowed to the child partitions, but they are allowed have a virtual view of the resources, in terms of virtual devices. Any request to the virtual devices is redirected to the devices in the parent partition, which then manages the requests [4]. The VMBus is a logical channel which enables inter-partition communication. The request and response are redirected via the VMBus. If the devices in the parent partition are also virtual devices, it will be redirected further until it reaches the parent partition, where it will gain access to the physical devices.

PREREQUISITES

To install Hyper-V we need to have an x64 based processor with a minimum of 1.4GHz clock speed. We also need to enable hardware-assisted virtualization, this feature is available in processors that include an inbuilt virtualization option specifically, Intel Virtualization Technology or AMD Virtualization. It also requires hardware-enforced Data Execution Prevention (DEP). Specifically, Intel XD bit (execute disable bit) or AMD NX bit (no execute bit)must be enabled [3]. The processor should also support second level address translation. Minimum 2 GB memory with error correcting code or similar technology is required, realistically much more memory is required as each virtual machine requires its own memory. The installation also requires a minimum of 32GB disk space. Apart from the above mentioned requirements, there are other requirements which are not mandatory but required to enable certain features [2].

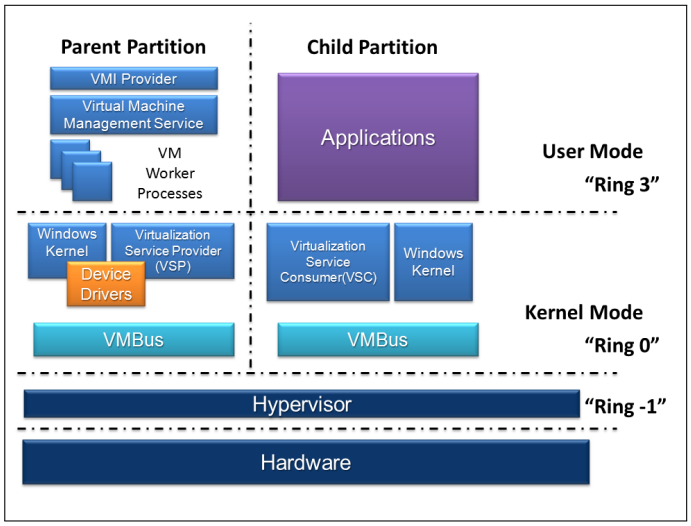


Fig. 1. Hyper-V architechture [2].

INSTALLATION

Installation of Hyper-V needs you to install Windows Server 2012 R2, for which you need a bootable device having the same and boot the device using it. Select the operating system you need to install i.e. standard/datacenter. Accept the terms and select install windows and select the drive you want to install the operating system on, we need a minimum of 32GB space for the installation. Set the username and password and then login. After logging on change the host name by going under my computer system properties. Open the server manager and configure according to the requirements which include selecting the roles and feature as required. Select the server name and click next and wait for the installation to complete [3]. Once Hyper-V is installed you can install create virtual networks and create virtual machines on which you can then install the operating system.

ADVANTAGES AND FEATURES

With Windows Server 2012, Hyper-V supports network virtualization, multi-tenancy, .vhdx disk format supporting virtual hard disks as large as 64TB, offloaded data transfer, cross-premises connectivity and Hyper-V replica. And with Windows Server 2012 R2 also supports shared virtual hard disk, storage quality of services, enhances session mode [2]. Multiple physical servers can be easily consolidated into comparatively fewer servers by implementing virtualization with Hyper-V. Consolidation accommodates the full use of deployed hardware resources. Hyper-V also helps in ease of administration as consolidation and centralization of resources simplifies administration and scale-up or scale-out can be accommodated with much greater ease. With Hyper-V and with virtualization, in general, there are significant cost savings. As separate physical machines are not required for every host and multiple virtual machines can be easily setup on a single physical machine. Hyper-V can be easily managed and a comprehensive Hyper-V management solution is available with System Center Virtual Machine Manager. Additional processing power, network bandwidth, and storage capacity can be accomplished quickly and easily by assigning additional resources from the host computer to the guest virtual machinws [5].

LIMITATIONS

Hyper-V does not virtualize audio hardware. It does not support the host/root operating system's optical drives to pass-through guest VMs, as a result, burning to disks are not supported. In Windows Server 2008, Hyper-V does not support live migration of guest VMs where live migration is maintaining network connections and uniterrupted services during VM migration between physical hosts. Although Hyper-V doesn't provide live migration but it tries to eliminate the limitation by having quick migration feature. Also, when Hyper-V is installed it uses VT-x x86 virtualization feature making it unavailable for other solutions due to which software which requires VT-x support can't be installed in parallel [2]. One of the major operating system that is still unsupported includes Fedora 8 and 9.

MANAGEMENT

Hyper-V servers can be managed using Windows PowerShell either locally or remotely. By running server manager on a remote computer, a server running in server core mode can be connected. It can also be connected using Microsoft Management Console (MMC) snap-in or by using another computer running Windows, the user can use Remote Desktop Services to run scripts and tools on a server. Server can be switched to GUI mode to use the usual user interface tools to accomplish the tasks and then switch back to server core mode [6].

Hype-V hosts can be managed using Hyper-V manager where the manager lets you manage a small number of Hyper-V hosts, both remote and local. It's gets installed with the installation of Hyper-V Management Tools, which can be installed through a full Hyper-V installation or a tools-only installation [6].

COMPARISION BETWEEN HYPER-V AND VSPHERE

Windows Serven 2012 R2, VMware vSphere Hypervisor and VMware vSphere 5.5 Enterprise Plus all support 320 logical processors, 4TB of physical memory, 1TB of memory per virtual machine. While Hyper-V and vSphere Enterprise edition both support 64 virtual CPUs per virtual machine, vSphere Hypervisor only support 8. In Hyper-V there can be 1,024 active VMs per host while this is limited to 512 in vSphere. It is interesting to know that Hyper-V supports up to 64 nodes and up to 8,000 virtual machines per in a cluster while vShere Enterprise plus supports 32 and 4,000 respectively [1].

Both VMware vSphere Hypervisor and Hyper-V are free standalone hypervisors, however enterprise edition of vSphere is not. The above information shows that Hyper-V has a number of advantages from a scalability perspective, especially when it comes to comparison with the vSphere Hypervisor [1]. VMware vSphere 5.5 brought a number of scalability increases for vSphere environments, doubling the number of host logical processors supported from 160 to 320, and doubling the host physical memory from 2TB to 4TB, but this still only brings vSphere up to the level that Hyper- V has been offering since September 2012 [1]. Hyper-V also supports double the number of active virtual machines per host, than both the vSphere Hypervisor and vSphere 5.5 Enterprise Plus.

CONCLUSION

Hyper-V is a powerful hypervisor introduced by Microsoft with features such as high availability, scalability, reliability, flexibility.

It also supports resouce monitoring that helps user track historical data on the use of virtual machines and gain insight into the resouce use of specific servers. Hyper-V sees competition from many other supervisors such as vSphere, Qemu, KVM, VirtualBox and provides a tough competition. Hyper-V requires hardware assisted virtualization support from processors and it can only be used wtih x86-64 processors. In spite of it's limitations it's a popular choce because of the features it provides. The customization options available in Hyper-V provides the user with lot of options to manage the virtual machines as he would like. Hyper-V and Hyper-V hosts can be easily managed using Windows PowerShell and Hyper-V manager tool locally or remotely.

REFERENCES

[1] Microsoft, "Why Hyper-V?" Web Page, Mar. 2016, accessed: 2017-03-22. [Online]. Available: https://download.microsoft.com/download/E/8/E/E8ECBD78-F07A-4A6F-9401-AA1760ED6985/Competitive-Advantages-of-Windows-Server-Hyper-V-over-VMware-vSphere.pdf

[2] Wikipedia, "Hyper-V," Web Page, Mar. 2016, accessed: 2017-03-21. [Online]. Available: https://en.wikipedia.org/wiki/Hyper-V

[3] Microsoft, "Hyper-V Configuration Guide," Web Page, Mar. 2017, accessed: 2017-03-20. [Online]. Available: https://gallery.technet.microsoft.com/Hyper-v-Step-by-step-84632942/file/122329/1/Hyper-vConfigGuide.pdf

[4] Microsoft, "Hyper-V Architecture," Web Page, Mar. 2016, accessed: 2017-03-21. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc768520(v=bts.10).aspx

[5] Microsoft, "Hyper-V Feature Overview," Web Page, Mar. 2017, accessed: 2017-03-24. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc768521(v=bts.10).aspx

[6] Microsoft, "Remotely manage Hyper-V hosts with Hyper-V Manager," Web Page, Mar. 2017, accessed: 2017-03-23. [Online]. Available: https://technet.microsoft.com/en-us/windows-server-docs/compute/hyper-v/manage/remotely-manage-hyper-v-hosts

Retainable Evaluator Execution Framework

PRATIK JAIN

School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

Corresponding authors: jainps@iu.edu

Paper-2, March 26, 2017

Apache REEF is a Big Data system that makes it easy to implement scalable, fault-tolerant runtime environments for a range of data processing models on top of resource managers such as Apache YARN and Mesos. The key features and abstractions of REEF are discussed. Two libraries of independent value are introduced. Wake is an event-based-programming framework and Tang is a dependency injection framework designed specifically for configuring distributed systems.

Keywords: REEF, Tang, Wake

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2012/report.pdf

INTRODUCTION

With the continuous growth of Hadoop the range of computational primitives expected by its users has also broadened. A number of performance and workload studies have shown that Hadoop MapReduce is a poor fit for iterative computations, such as machine learning and graph processing. Also, for extremely small computations like ad-hoc queries that compose the vast majority of jobs on production clusters, Hadoop MapReduce is not a good fit. Hadoop 2 addresses this problem by factoring MapReduce into two components: an application master that schedules computations for a single job at a time, and YARN, a cluster resource manager that coordinates between multiple jobs and tenants. In spite of the fact that this resource manager, YARN, allows a wide range of computational frameworks to coexist in one cluster, many challenges remain [1].

From the perspective of the scheduler, a number of issues arise that must be appropriately handled in order to scale-out to massive datasets. First, each map task should be scheduled close to where the input block resides, ideally on the same machine or rack. Second, failures can occur at the task level at any step, requiring backup tasks to be scheduled or the job being aborted. Third, performance bottlenecks can cause an imbalance in the task-level progress. The scheduler must react to these stragglers by scheduling clones and incorporating the logical task that crosses the finish line first.

Apache REEF (Retainable Evaluator Execution Framework), a library for developing portable applications for cluster resource managers such as Apache Hadoop YARN or Apache Mesos, addresses these challenges. It provides a reusable control-plane for scheduling and coordinating task-level work on cluster resource managers. The REEF design enables sophisticated optimizations, such as container re-use and data caching, and facilitates workflows that span multiple frameworks. Examples include pipelining data between different operators in a relational system, retaining state across iterations in iterative or recursive data flow, and passing the result of a MapReduce job to a Machine Learning computation.

FEATURES

Due to its following critical features, Apache REEF drastically simplifies development of resource managers [2].

Centralized Control Flow

Apache REEF turns the chaos of a distributed application into various events in a single machine. These events include container allocation, Task launch, completion, and failure.

Task runtime

Apache REEF provides a Task runtime which is instantiated in every container of a REEF application and can keep data in memory in between Tasks. This enables efficient pipelines on REEF.

Support for multiple resource managers

Apache REEF applications are portable to any supported resource manager with minimal effort. In addition to this, new resource managers are easy to support in REEF.

.NET and Java API

Apache REEF is the only API to write YARN or Mesos applications in .NET. Additionally, a single REEF application is free to mix and match tasks written for .NET or Java.

Plugins

Apache REEF allows for plugins to augment its feature set without hindering the core. REEF includes many plugins, such as a name-based communications between Tasks, MPI-inspired group communications, and data ingress.

As a result of such features Apache REEF shows properties like retainability of hardware resources across tasks and jobs, composability of operators written for multiple computational frameworks and storage backends, cost modeling for data movement and single machine parallelism, fault handling [3] and elasticity.

KEY ABSTRACTIONS

REEF is structured around the following key abstractions [4]:

Driver: This is a user-supplied control logic that implements the resource allocation and Task scheduling logic. There is exactly one Driver for each Job. The duration and characteristics of the Job are determined by this module.

Task: This encapsulates the task-level client code to be executed in an Evaluator.

Evaluator: This is a runtime environment on a container that can retain state within Contexts and execute Tasks (one at a time). A single evaluator may run many activities throughout its lifetime. This enables sharing among Activities and reduces scheduling costs.

Context: It is a state management environment within an Evaluator that is accessible to any Task hosted on that Evaluator.

Services: Objects and daemon threads that are retained across Tasks that run within an Evaluator [1]. Examples include caches of parsed data, intermediate state, and network connection pools.

WAKE AND TANG

The lower levels of REEF can be decoupled from the data models and semantics of systems built atop it. This results in two standalone systems, Tang and Wake which are both language independent and allow REEF to bridge the JVM and .NET.

Tang is a configuration management and checking framework [5]. It emphasizes explicit documentation and automatic checkability of configurations and applications instead of ad-hoc, application-specific configuration and bootstrapping logic. It not only supports distributed, multi-language applications but also gracefully handles simpler use cases. It makes use of dependency injection to automatically instantiate applications. Given a request for some type of object, and information that explains how dependencies between objects should be resolved, dependency injectors automatically instantiate the requested object and all of the objects it depends upon. Tang makes use of a few simple wire formats to support remote and even cross-language dependency injection.

Wake is an event-driven framework based on ideas from SEDA, Click, Akka and Rx [6]. It is general purpose in the sense that it is designed to support computationally intensive applications as well as high-performance networking, storage, and legacy I/O systems. Wake is implemented to support high-performance, scalable analytical processing systems i.e. big data applications. It can be used to achieve high fanout and low latency as well as high-throughput processing and it can thus aid to implement control plane logic and the data plane.

Wake is designed to work with Tang. This makes it extremely easy to wire up complicated graphs of event handling logic. In addition to making it easy to build up event-driven applications, Tang provides a range of static analysis tools and provides a simple aspect-style programming facility that supports Wake's latency and throughput profilers.

RELATIONSHIPS WITH OTHER APACHE PRODUCTS

Given REEF's position in the big data stack, there are three relationships to consider: Projects that fit below, on top of, or alongside REEF in the stack [**?**].

Below REEF

REEF is designed to facilitate application development on top of resource managers like Mesos and YARN. Hence, its relationship with the resource managers is symbiotic by design.

On Top of REEF

Apache Spark, Giraph, MapReduce and Flink are some of the projects that logically belong at a higher layer of the big data stack than REEF. Each of these had to individually solve some of the issues REEF addresses.

Alongside REEF

Apache builds library layers on top of a resource management platform. Twill, Slider, and Tez are notable examples in the incubator. These projects share many objectives with REEF. Twill simplifies programming by exposing a programming model. Apache Slider is a framework to make it easy to deploy and manage long-running static applications in a YARN cluster. Apache Tez is a project to develop a generic Directed Acyclic Graph (DAG) processing framework with a reusable set of data processing primitives. Apache Helix automates application-wide management operations which require global knowledge and coordination, such as repartitioning of resources and scheduling of maintenance tasks. Helix separates global coordination concerns from the functional tasks of the application with a state machine abstraction.

CONCLUSION

REEF is a flexible framework for developing distributed applications on resource manager services. It is a standard library of reusable system components that can be easily composed into application logic. It possesses properties of retainability, composability, cost modeling, fault handling and elasticity. Its key components are driver, task, evaluator, context and services. Its relationship with projects above it, below it and alongside it in the big data stack is discussed. Thus, a brief overview of REEF is shown here.

ACKNOWLEDGEMENTS

The author thanks Prof. Gregor von Laszewski and all the course AIs for their continuous technical support.

REFERENCES

[1] Byung-Gon Chun, Chris Douglas, Shravan Narayanamurthy, Josh Rosen, Tyson Condie, Sergiy Matusevych, Raghu Ramakrishnan, Russell Sears, Carlo Curino, Brandon Myers, Sriram Rao, Markus

Weimer, “Reef: Retainable evaluator execution framework,” in *VLDB Endowment, Vol. 6, No. 12*, Aug. 2013. [Online]. Available: http://db.disi.unitn.eu/pages/VLDBProgram/pdf/demo/p841-sears.pdf

[2] The Apache Software Foundation, “Apache reef™ - a stdlib for big data,” Web Page, Nov. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/

[3] Techopedia Inc., “Retainable evaluator execution framework (reef),” Web Page, Jan. 2017, accessed 2017-03-17. [Online]. Available: https://www.techopedia.com/definition/29891/retainable-evaluator-execution-framework-reef

[4] Markus Weimer, Yingda Chen, Byung-Gon Chun, Tyson Condie, Carlo Curino, Chris Douglas, Yunseong Lee, Tony Majestro, Dahlia Malkhi, Sergiy Matusevych, Brandon Myers, Shravan Narayanamurthy, Raghu Ramakrishnan, Sriram Rao, Russell Sears, Beysim Sezgin, Julia Wang, “Reef: Retainable evaluator execution framework,” in *Proc ACM SIGMOD Int Conf Manag Data. Author manuscript*, Jan. 2016, pp. 1343–1355. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4724804/

[5] The Apache Software Foundation, “Tang,” Web Page, Nov. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/tang.html

[6] ——, “Wake,” Web Page, Dec. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/wake.html

An Overview of Apache Avro

ANVESH NAYAN LINGAMPALLI[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: anveling@umail.iu.edu

March 26, 2017

Apache Avro is a data serialization system, which uses JSON based schemas and RPC calls to send the data. Hadoop based tools natively support Avro for serialization and deserialization of the data.

Keywords: apache, avro, serialization

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2016/report.pdf

1. INTRODUCTION

Apache Avro [1] is a data serialization system. Data serialization is a mechanism to translate data in a computer environment, such as memory buffer, data structures, object state into binary or textual form. Java and Hadoop provides serialization APIs, which are java based. Apache Avro is a language independent system, that can be processed by multiple languages [2].

Avro is heavily dependent on schemas. These schemas are defined with JSON that simplifies its implementations with its libraries. Different schemas can be used for serialization and deserialization, and Avro will handle the missing fields or extra fields [3]. When Avro data is stored in a file, its schema is also stored with it. Avro stores the data definition in JSON format, which makes it easier to interpret.

Apache Avro provides rich data structures, compact binary data format, a container file, facility for remote procedure calls (RPC) and integration with dynamic languages [1]. In Remote procedure calls, the client and server exchange schemas during the connection. This exchange helps when there are missing fields or extra fields. Avro works well with Hadoop MapReduce, which provides the large scale processing across many processors to do the calculations simultaneously and efficiently.

2. COMPONENTS OF AVRO

Avro has two main components, data serialization and remote procedure call (RPC) support.[4]

2.1. Data Serialization

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. To serialize data using Avro, 1) define an Avro Schema. 2) compile the schema using Avro utility. 3) Java code corresponding to that schema is obtained. 4) populate the schema with data. 5) serialize the data using Avro library[**?**].

2.2. Remote Procedure Call

In a Remote Procedure Call, the client and server exchange schemas in the connection handshake. (This can be optimized so that, for most calls, no schemas are actually transmitted.) Since both client and server both have the other's full schema, correspondence between same named fields, missing fields, extra fields, etc. can all be easily resolved[5].

3. SPECIFICATIONS

3.1. Encoding

Avro specifies two serialization encodings, binary and JSON. Binary encoding is smaller and faster, but not efficient in the case of debugging and web-based applications, where JSON encoding is appropriate[6].

3.2. Object Container Files

Avro includes an object container file format. Objects stored in blocks that may be compressed and syncronization markers are used to permit splitting of the files for MapReduce processing. File consists of, a header followed by one or more data blocks. Header consists of four bytes 'O', 'b', 'j', 1. It also consists of the file metadata, including the schema and a 16-byte, randomly generated sync marker[7]. Currently used file metadata properties are, avro.schema which contains the schema of the objects stored in file avro.codec contains the name of the compression codec used to compress the blocks.

A file data block consists of, count of the objects in the block, size of bytes of the serialized objects in current block, the serialized objects and a 16-byte sync marker[7].

Each block's binary data can be efficiently extracted without deserializing the contents. The combination of the block size, object counts, and sync markers enable detection of corrupt blocks.

4. KEY FEATURES OF APACHE AVRO

Apache Thrift and Google's Protocol buffers are the competent libraries with Apache Avro. But Avro is fundamentally different from these frameworks. The key features of the Apache Avro are, Dynamic typing, untagged data, no manually-assigned field IDs.

4.1. Dynamic Typing

Serialization and deserialization without the code generation is possible in Apache Avro. Data is always accompanied by a schema that permits full processing of the data. This feature of the Avro, makes it possible to construct data-processing systems and languages. Avro creates binary structure format that is both compressible and splittable [3].

4.2. Untagged Data

Binary data with a schema together, allows the data to be written without any overhead. This feature provides faster data processing and compact data encoding.

4.3. Schema Evolution

This feature of Avro, is essentially a robust way to support various data schemas that change over time. Avro cleanly handles schema changes such as missing fields, added fields and changed fields. By using this feature, new programs can read old data and old programs can read new data.

5. APPLICATION

Primary use of Apache Avro is in Apache Hadoop where it can provide both serialization format for persistent data, and a wire format for communication between Hadoop nodes.

Avro was designed for Hadoop for making it interoperable across different languages. With Avro serialization, Pig utilizes AvroStorage().

5.1. Using Avro with Eventlet

Eventlet[4] is a concurrent networking library for python that allows the changing of the code, to run the code, instead of writing it. RPC support from the Avro combined with Eventlet is used for building highly concurrent network-based services. Avro is present in the transport layer on top of HTTP for RPC calls. It POSTS binary data to the server and processes the response. Eventlet.wsgi (Web server Gateway Interface) is used to build RPC server.

6. DISADVANTAGES

Avro is not the fastest in serialization process, but it is one of the faster frameworks. The syntax of Avro can be error prone. And error handling is complex whn compared with Protocol buffers and Thrift.

7. EDUCATIONAL MATERIAL

Tutorialspoint.com [3]for Apache Avro provides the resources to use Avro for deserialization and serialization of the data. Apache Avro's [1] website provides documentation for understanding, deploying and managing the framework.

8. CONCULSION

Apache Avro is a framework, which allows the serialization of data that has schema built in it. The serialization of data is results in compact binary format, which does require proxy objects. Instead of using generated proxy object libraries and strong typing, Avro heavily relies on the schema that are sent along with serialized data.

REFERENCES

[1] "Apache avro," Web Page, accessed: 2017-03-21. [Online]. Available: https://avro.apache.org/

[2] "Apache Avro documentation," Web Page, accessed: 2017-03-21. [Online]. Available: https://avro.apache.org/docs/1.7.7/index.html

[3] "Avro tutorial," Web Page, accessed: 2017-03-22. [Online]. Available: https://www.tutorialspoint.com/avro/avro_overview.html

[4] "Using Avro with Eventlet," Web Page, accessed: 2017-03-22. [Online]. Available: http://unethicalblogger.com/2010/05/07/how-to-using-avro-with-eventlet.html

[5] "Rpc by avro," Web Page, accessed: 2017-03-22. [Online]. Available: https://www.igvita.com/2010/02/16/data-serialization-rpc-with-avro-ruby/

[6] "Encoding in avro," Web Page, accessed: 2017-03-22. [Online]. Available: https://avro.apache.org/docs/1.8.1/spec.html

[7] "Apache avro wiki," Web Page, accessed: 2017-03-22. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Avro

An Overview of Pivotal HD/HAWQ and its Applications

VEERA MARNI[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: vmarni@umail.iu.edu*

March 25, 2017

Pivotal HD/HAWQ is a Hadoop native SQL query engine that combines the key technological advantages of MPP database with the scalability and convenience of Hadoop.

Keywords: pivotal, hd, hawq, data science

https://github.com/narayana1043/sp17-i524/blob/master/paper2/S17-IR-2017/report.pdf

1. INTRODUCTION

Pivotal-HAWQ[1] is a Hadoop native SQL query engine that combines the key technological advantages of MPP database with the scalability and convenience of Hadoop. HAWQ reads data from and writes data to HDFS natively. HAWQ delivers industry-leading performance and linear scalability. It provides users the tools to confidently and successfully interact with petabyte range data sets. HAWQ provides users with a complete, standards compliant SQL interface.

An MPP database is a database that is optimized to be processed in parallel for many operations to be performed by many processing units at a time.[2] HAWQ breaks complex queries into small tasks and distributes them to MPP query processing units for execution. HAWQ's basic unit of parallelism is the segment instance. Multiple segment instances on commodity servers work together to form a single parallel query processing system. A query submitted to HAWQ is optimized, broken into smaller components, and dispatched to segments that work together to deliver a single result set. All relational operations - such as table scans, joins, aggregations, and sorts - simultaneously execute in parallel across the segments. Data from upstream components in the dynamic pipeline are transmitted to downstream components through the scalable User Datagram Protocol (UDP) interconnect.

Based on Hadoop's distributed storage, HAWQ has no single point of failure and supports fully-automatic online recovery. System states are continuously monitored, therefore if a segment fails, it is automatically removed from the cluster. During this process, the system continues serving customer queries, and the segments can be added back to the system when necessary.

2. ARCHITECTURE OF HAWQ

In a typical HAWQ deployment, each slave node has one physical HAWQ segment, an HDFS DataNode and a NodeManager installed. Masters for HAWQ, HDFS and YARN are hosted on separate nodes.[3]. HAWQ is tightly integrated with YARN, the Hadoop resource management framework, for query resource management. HAWQ caches containers from YARN in a resource pool and then manages those resources locally by leveraging HAWQ's own finer-grained resource management for users and groups.

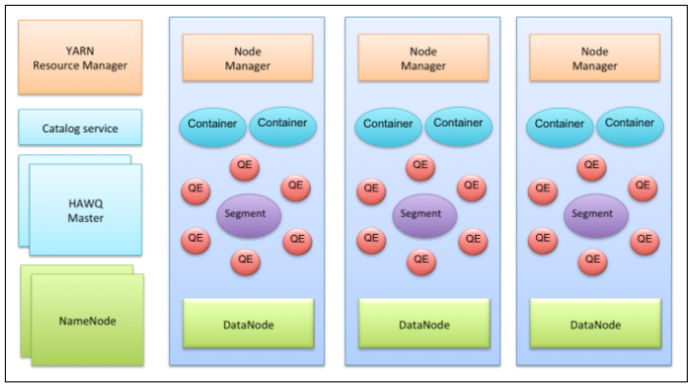


Fig. 1. The following diagram provides a high-level architectural view of a typical HAWQ deployment.[4].

2.1. HAWQ Master

The HAWQ master is the entry point to the system. It is the database process that accepts client connections and processes the SQL commands issued. The HAWQ master parses queries, optimizes queries, dispatches queries to segments and coordinates the query execution. End-users interact with HAWQ through the master and can connect to the database using client programs such as psql or application programming interfaces (APIs) such as JDBC or ODBC.The master is where the global system catalog resides. The global system catalog is the set of system tables that contain metadata about the HAWQ system

itself. The master does not contain any user data; data resides only on HDFS.

2.2. HAWQ Segment

In HAWQ, the segments are the units that process data simultaneously. There is only one physical segment on each host. Each segment can start many Query Executors (QEs) for each query slice. This makes a single segment act like multiple virtual segments, which enables HAWQ to better utilize all available resources.

2.3. HAWQ Interconnect

The interconnect is the networking layer of HAWQ. When a user connects to a database and issues a query, processes are created on each segment to handle the query. The interconnect refers to the inter-process communication between the segments, as well as the network infrastructure on which this communication relies.

2.4. HAWQ Resource Manager

The HAWQ resource manager obtains resources from YARN and responds to resource requests. Resources are buffered by the HAWQ resource manager to support low latency queries. The HAWQ resource manager can also run in standalone mode. In these deployments, HAWQ manages resources by itself without YARN.

2.5. HAWQ Catalog Service

The HAWQ catalog service stores all metadata, such as UDF/UDT[5] information, relation information, security information and data file locations.

2.6. HAWQ Fault Tolerance Service

The HAWQ fault tolerance service (FTS) is responsible for detecting segment failures and accepting heartbeats from segments.

2.7. HAWQ Dispatcher

The HAWQ dispatcher dispatches query plans to a selected subset of segments and coordinates the execution of the query. The dispatcher and the HAWQ resource manager are the main components responsible for the dynamic scheduling of queries and the resources required to execute them.

3. KEY FEATURES OF PIVOTAL HDB/HAWQ

3.1. High-Performance Architecture

Pivotal HDB's parallel processing architecture delivers high performance throughput and low latency (potentially, near-real-time) query responses that can scale to petabyte-sized datasets. Pivotal HDB also features a cutting-edge, cost-based SQL query optimizer and dynamic pipelining technology for efficient performance operation.

3.2. Robust ANSI SQL Compliance

Pivotal HDB complies with ANSI SQL-92, -99, and -2003 standards, plus OLAP extensions. Leverage existing SQL expertise and existing SQL-based applications and BI/data visualization tools. Execute complex queries and joins, including roll-ups and nested queries.

3.3. Deep Analytics and Machine Learning

Pivotal HDB integrates statistical and machine learning capabilities that can be natively invoked from SQL and applied natively to large data sets across a Hadoop cluster. Pivotal HDB supports PL/Python, PL/Java and PL/R programming languages.

3.4. Flexible Data Format Support

HDB supports multiple data file formats including Apache Parquet and HDB binary data files, plus HBase and Avro via HDB's Pivotal Extension Framework (PXF) services. HDB interfaces with HCatalog, which enables you to query an even broader range of data formats.

3.5. Tight Integration with Hadoop Ecosystem

Pivotal HDB plugs into the Apache Ambari installation, management and configuration framework. This provides a Hadoop-native mechanism for installation and deployment of Pivotal HDB and for monitoring cluster resources across Pivotal HDB and the rest of the Hadoop ecosystem.

4. ECOSYSTEM

HAWQ uses Hadoop ecosystem[6] integration and manageability and flexible data-store format support. HAWQ is natively in hadoop and requires no connectors.Hadoop Common contains libraries and utilities needed by other Hadoop modules. HDFS is a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster. Hadoop YARN is a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users applications. Hadoop MapReduce is an implementation of the MapReduce programming model for large scale data processing.

5. APPLICATIONS OF PIVOTAL HD/HAWQ

The Pivotal HD Enterprise product enables you to take advantage of big data analytics without the overhead and complexity of a project built from scratch. Pivotal HD Enterprise is Apache Hadoop that allows users to write distributed processing applications for large data sets across a cluster of commodity servers using a simple programming model.

5.1. Content-Based Image Retrieval using Pivotal HD with HAWQ

Manual tagging is infeasible for image databases of this size, and is prone to errors due to users' subjective opinions.Given a query image, a CBIR[7] system can be potentially used to auto-tag (label) similar images in the collection, with the assigned label being the object category or scene description label. This technology also has an important role to play within a number of non-consumer domains. CBIR systems can be used in health care contexts for case-based diagnoses. A common example is image retrieval on large image databases such as Flickr[8].

5.2. Transition of Hulu from Mysql to Pivotal-HD/HAWQ

Hulu is a leading video company that offers TV shows, clips, movies and more on the free, ad-supported Hulu.com service and the subscription service Hulu Plus. It serves 4 billion videos. It has used HAWQ to gain performance improvement to handle queries from users. It's main challenge was inability to scale MySQL and Memcached to improve performance which was handled by Pivotal HAWQ[9].

6. DISADVANTAGES

There are also some drawbacks that needs attention before one choose to use Pivotal-HD/HAWQ. Since most of these are used with the help public cloud providers there is a greater dependency on service providers, Risk of being locked into proprietary or vendor-recommended systems, Potential privacy and security risks of putting valuable data on someone else's system. Another important problem what happens if the supplier suddenly stops services. Even with this disadvantages the technology is still used greatly in various industries and many more are looking forward to move into cloud.

7. EDUCATIONAL MATERIAL

Pivotal offers an portfolio of role-based courses and certifications to build your product expertise[10]. These courses can get someone with basic knowledge of hadoop ecosystem to understand how to deploy, manage, build, integrate and analyze Pivotal HD/HAWQ applications on clouds.

8. CONCULSION

Pivotal HD/HAWQ is the Apache Hadoop native SQL database for data science and machine learning workloads. With Pivotal HDB we can ask more questions on data in Hadoop to gain insights into data by using all the available cloud resources without sampling data or moving it into another platform for advanced analytics. Its fault tolerant architecture can handle node failures and move the workload around clusters.

REFERENCES

[1] "About hawq," webpage. [Online]. Available: http://hdb.docs.pivotal.io/212/hawq/overview/HAWQOverview.html

[2] M. Rouse and M. Haughn, "About mpp," webpage. [Online]. Available: http://searchdatamanagement.techtarget.com/definition/MPP-database-massively-parallel-processing-database

[3] "Architecture of pivotal hawq," webpage. [Online]. Available: http://hdb.docs.pivotal.io/212/hawq/overview/HAWQArchitecture.html

[4] "Pivotal architecture online image," webpage. [Online]. Available: https://hdb.docs.pivotal.io/211/hawq/mdimages/hawq_high_level_architecture.png

[5] "About udt," webpage. [Online]. Available: https://en.wikipedia.org/wiki/UDP-based_Data_Transfer_Protocol

[6] "About hadoop," webpage. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Hadoop

[7] "About cbir," webpage. [Online]. Available: https://content.pivotal.io/blog/content-based-image-retrieval-using-pivotal-hd-with-hawq

[8] E. Hörster, R. Lienhart, and M. Slaney, "Image retrieval on large-scale image databases," in *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, ser. CIVR '07. New York, NY, USA: ACM, 2007, pp. 17–24. [Online]. Available: http://doi.acm.org/10.1145/1282280.1282283

[9] "pivotal hulu use case," webpage. [Online]. Available: http://hdb.docs.pivotal.io/212/hawq/overview/HAWQArchitecture.html

[10] "educational courses," webpage. [Online]. Available: https://pivotal.io/training/courses

An overview of Cisco Intelligent Automation for Cloud

BHAVESH REDDY MERUGUREDDY[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: bmerugur@umail.iu.edu*

Paper2, March 27, 2017

Cisco Intelligent automation for cloud is a cloud platform that can deliver services across mixed environments. Its services range from underlying infrastructure to anything-as-a-service and allows the users to evaluate, transform and deploy various IT services. It provides a foundation for organizational transformation by expanding the uses of cloud technology beyond its infrastructure.

Keywords: Automation, Multitenancy, Integration, Orchestration, Provisioner

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2018/report.pdf

INTRODUCTION

Cisco Intelligent Automation for Cloud (IAC) provides a framework that allows users to make use of the cloud services effectively and manage the cloud beyond Infrastructure-as-a-service (IaaS) [1]. It can be considered as a unified cloud platform that can deliver any type of service across mixed environments [2]. This leads to an increase in cloud penetration across different businesses. Its services range from underlying infrastructure to anything-as-a-service by allowing its users to evaluate, transform and deploy the IT and business services in a way they desire. Cisco Intelligent Automation for Cloud automates sophisticated data center from a single self-service portal which is beyond the provision of virtual machines. It creates a catalog of standardized service offerings, thereby implementing policy-based controls. It also provides resource management across different aspects of IT infrastructure such as network, virtualization, storage, compute and applications.

COMPONENTS

Cisco Intelligent Automation for Cloud consists of major interface and automation components. In other words, it provides an integrated stack of core elements namely Cisco Cloud Portal, Cisco Process Orchestrator, Cisco Process Orchestrator Integration Framework, Cisco Server Provisioner, and Cloud Automation Packs [3].

Cisco Cloud Portal

Cisco Cloud Portal is a web-based service portal that helps users to order and manage services. It provides a configurable interface for different roles and departments which include managers, administrators and consumers [3]. This provides an access to a catalog of on-demand IT resources. Service requests for resources running on cross-platform infrastructure can be managed effectively by Cisco Cloud Portal. It tracks and manages lifecycle of each service from the initial request to withdrawal. The modules which constitute Cisco Cloud Portal are Cisco Portal Manager, Cisco Service Catalog, Cisco Request Center and Cisco Service Connector.

Cisco Portal Manager combines data from multiple sources providing a highly flexible portal interface. It manages the interface and increases user satisfaction. It provides a drag-and-drop facility on the interface making it easier for the users to create their own portal views which ensures flexibility [4]. Depending on the user requests, Cisco Service Catalog provides a controlled access to IT resources through standardized service options. It makes use of reusable components and tools for publishing services in the portal. Cisco Request Center provides lifecycle management and request management for the infrastructure services. To ensure data center management, it maintains policy-based controls and reduces the cycle time for the ordering, approval and provisioning process. It uses advanced methods for simplifying the ordering process. It also helps in streamlining end-to-end service delivery cycle time. Cisco Service Connector is a platform that can be used for integrating with third-party systems. This supports a heterogeneous data center environment by providing adaptors for integrating with automated provisioning systems.

Cisco Process Orchestrator

Cisco Process Orchestrator is a component of Cisco IAC responsible for automation of service management and assurance instantiation. It is an orchestration engine that provides an automation design studio and a reporting and analytics module. It is useful in automating and standardizing IT processes in heterogeneous environments [5]. It considers IT automation as a service-oriented approach and focuses more on services. It allows users to deploy services by defining new instances in

real time. Cisco Process Orchestrator acts as a foundation for building application, network and data center oriented solutions. It includes auditing and extensive reporting and provides workspaces for developers and administrators making it easier for the stakeholders to manage services.

The primary function of Cisco Process Orchestrator is to provide automation through integration with domain managers and tools in the environment [6]. Automation Packs and Adapters are the features used in the integration. Some of the services include event correlation, application provisioning and event application provisioning. It usually receives events requiring further analysis. It also includes security tools, configuration tools, change management systems, visualization management tools, provisioning systems and service desks.

Cisco Process Orchestrator Integration Framework

Cisco Process Orchestrator Integration Framework is responsible for integrating Cisco IAC with any data center element in the environment. It connects IT service management tools into streamlined and automated processes by making use of field-built integration. VMware, SAP, Oracle DB, Remedy and Windows are some of the available integrations. Field integration and automation are carried out by the design studio which provides web services, database access and command-line interface.

Cisco Server Provisioner

Cisco Server Provisioner is a software application used for deploying systems with Linux, Windows and Hypervisors from bare metal in IT organizations and data centers [7]. It acts as an application for native and remote installations on physical and virtual servers. It provides imaging component for OS and Hypervisor. The Cisco Server Provisioner can be considered as a server suite that consists of Bare Metal Provisioning Payloads and Bare Metal Imaging Payloads which provide GUI and API interfaces including remote file copying and remote troubleshooting. Provisioner runs on a dedicated system which does not run any other application. Some of the benefits of the Provisioner include reduced deployment time and increased utilization of systems.

Cloud Automation Packs

Cloud Automation Packs are the workflows useful for complex computing tasks. The tasks include automation of core activities that cover various domains, Cisco Server Provisioner task automation, VMware task automation and Cisco UCS Manager task automation. Automation Packs provide a set of target groups, variables, configurations and process definitions required for defining automated IT processes. Automation Packs combine with Adaptors to enable integrations. Integration with IT element is carried out through a combination of automation content from an Automation Pack and a set of Adaptors. Some of the integration scenarios include Command Line Interface invocations, Web Service integration, Messaging integration and Scripting support. Automation content can leverage Adaptors to enable these scenarios. Automation Packs can be used to build, pack, update and ship the integrations.

FUNCTIONS

Cisco Intelligent Automation for Cloud provides a platform for designing, deploying and operating a cloud infrastructure in a public, private or a hybrid model. It supports various cloud management activities including setup and design, system operations, reporting and analytics.

Self-Service Interface

Self-Service Interface is a web-based interface which allows the users to view the service catalog according to their roles and other access controls. It provides dynamic forms by which users can provide configuration details and order services. It also allows the users to track order status, manage and modify placed orders and view the usage and consumption.

Service Delivery Automation

After the approval of the placed orders, Service Delivery Automation takes place to orchestrate the configuration and provision of resources like compute, network, storage and supporting services such as firewall, disaster recovery and load balancing [3]. The automated provisioning provides consumption tracking and integration into metering and billing systems. The automated processes then orchestrate the configuration updates and allows the service information updates to be sent back to the system management tools and web-based portal.

Operational Process Automation

Operational Process Automation coordinates the operational tasks for cloud management which include service-level management, alerting, reporting, capacity planning, performance management, user management and maintenance checks. It provides user administration capability to control user roles and identity, placing the users securely isolated from each other. Systems incidents can be managed by the users through alert management feature and all the processes and results can be tracked and reported by the reporting functionality. Operational Process Automation consists of an Automation Control Center which is a console for viewing and controlling automated processes.

Network Automation

Cisco IAC allows a manual pre-provisioning of network layer with the increasing amount of data being placed in cloud. This is carried out by Network Automation. Network Automation enables deployment of network services through the Self-Service Portal with a single order [8]. It facilitates dynamic installation of virtual network devices with onboarding users and creates network topologies for the users based on the applications they use.

Cloud Governance

Through Cloud Governance, Cisco IAC delivers a set of measures for tracking business-oriented metrics. Cloud Governance provides portfolio management across different cloud environments. It allows organizations to establish limits ahead of time. Financial granularity is achieved by these consumption limits.

Advanced Multitenancy

Advanced Multitenancy allows multiple users to securely reside in a shared environment by providing isolated containers. User management work is offloaded from the cloud provider as the users are controlled by the cloud administrators. Multitenancy sets service pricing per user and accordingly, the services can be enabled or disabled per organization or user. It provides onboarding, modification and offboarding of users and enables secure containers by instantiating network devices.

Multicloud Management

By Multicloud Management, service providers can tailor their services to specific project needs and specific functions of the hypervisor platforms. For example, Cisco IAC supports two infrastructure layers namely Cisco UCS Director and OpenStack [8]. It allows administrators to manage network services and virtual machines by integrating with Havana and Icehouse. This allows the Cisco UCS Director users to manage Microsoft System Center Virtual Machine Manager.

Resource Management

Cisco IAC Resource Management orchestrates resource-level operations across different hypervisors such as Hyper-V, Xen or VMware, compute resources such as Cisco UCS, network resources and storage resources such as NetApp. This is done by orchestrating the requests to domain resource managers. It provides maintenance and replacement of units. Capacity management is provided by automated capacity utilization checks, trending reports and alerts. The usage and user quota are managed by automated monitoring and metering of user accounts.

Lifecycle Management

Lifecycle Management is responsible for creation of service definitions which include selection parameters, business and technical processing flows, pricing options and design descriptions. It also involves management of a service model and underlying automation design for managing a service. Automation design is managed by creating workflows to automate service provisioning, modification, decommissioning and upgrades. The designs are modified by point-and-click-tools instead of custom programming. Lifecycle Management tracks all aspects of services that are running which includes business and project information.

DEPLOYMENT

Cisco Intelligent Automation for Cloud is deployed as a software solution. For planning, preparation, design, implementation and optimization of cloud services, it is deployed along with services engagement. The services engagement involves creation of automation workflows and development of a cloud strategy. It focuses on service capabilities for the software deployment.

CONCLUSION

Cisco Intelligent Automation for Cloud is a framework useful in expanding cloud responsiveness and flexibility. It provides services beyond provisioning virtual machines. It delivers various services in a self-service manner across mixed environments. The cloud management in Cisco Intelligent Automation for Cloud allows users to evaluate, procure and deploy IT services according to the needs. Self-Service Portal, Service Delivery Automation, Network Automation, Resource Management, Provisioning, Advanced Multitenancy, Portfolio Management and Lifecycle Management are the elements responsible for the cloud management in Cisco Intelligent Automation for Cloud.

ACKNOWLEDGEMENTS

The author thanks Professor Gregor Von Lazewski and all the AIs of Big Data class for the guidance and technnical support.

45

REFERENCES

[1] T. Hagay, "Build a better cloud with cisco intelligent automation for cloud," Webpage, 2014. [Online]. Available: https://www.ciscolive.com/online/connect/sessionDetail.ww?SESSION_ID=76497&tclass=popup

[2] S. Miniman, "Cisco moves up the cloud stack with intelligent automation," Webpage. [Online]. Available: http://wikibon.org/wiki/v/Cisco_Moves_Up_the_Cloud_Stack_with_Intelligent_Automation

[3] *Cisco Intelligent Automation for Cloud*, Cisco Systems, 2011. [Online]. Available: http://www.cisco.com/c/dam/en_us/training-events/le21/le34/downloads/689/vmworld/CIAC_Cisco_Intelligent_Automation_Cloud.pdf

[4] *Cisco Cloud Portal*, Cisco Systems, 2011. [Online]. Available: http://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/cloud-portal/cisco_cloud_portal.pdf

[5] "Cisco process orchestrator," Webpage. [Online]. Available: http://www.cisco.com/c/en/us/products/cloud-systems-management/process-orchestrator/index.html

[6] *Cisco Process Orchestrator 3.0 Integrations and Automation Packs*, Cisco Systems, 2013. [Online]. Available: http://www.cisco.com/en/US/docs/net_mgmt/datacenter_mgmt/Process_Orchestrator/3.0/Integration_Automation/CPO_3.0_Integrations_and_Automation_Packs.pdf

[7] *Cisco Server Provisioner User's Guide*, LinMin Corp. and Cisco Systems. [Online]. Available: https://linmin.com/cisco/help/index.html?introduction.html

[8] F. Mondora, "Cisco intelligent automation for cloud 4.0," Webpage. [Online]. Available: https://www.mondora.com/#!/post/12d838f9aa9a6a3c144d29cc6ff56a7c

Amazon Elastic Beanstalk

SHREE GOVIND MISHRA[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: shremish@indiana.edu

project-000, March 26, 2017

Amazon Elastic Beanstalk is a service provided by the Amazon Web Services which allow developers and engineers to easily deploy and run web applications in the cloud, in such a way that these applications are highly available and scalable. Elastic Beanstalk manages the deployed application by reducing the management complexities as it automatically handles the capacity provisioning, load balancing, scaling, and application health monitoring. Elastic BeanStalk also provisions one or more AWS Resources such as Amazon EC2 instances when an application is deployed.

Keywords: Cloud, I524, Amazon Web Services, AWS Elastic Beanstalk, Load Balancing, Cloud Watch, AWS Management Console, Auto Scaling

ttps://github.com/Govind273/sp17/i524/blob/master/paper2/S17-IR-2021/report.pdf

INTRODUCTION

Cloud Computing can be defined as an abstraction of services from infrastructures(i.e hardware), platforms and applications(i.e software) by virtualization of resources [1]. The different form of cloud computing services include IaaS, PaaS, and SaaS which stands for (Infrastructure as a Service), Platform as a Service, and Software as a service respectively. With Cloud Computing, organizations can consume shared computing and storage resources rather than building, operating and improving infrastructure on their own and it can also enable organizations to obtain a flexible, secure and cost-effective Infrastructure.

Amazon Web Services (AWS) is a subsidiary of Amazon.com, which offers a suite of cloud computing services such as compute power, database storage, content delivery and other functionalities that make up an on-demand computing platform. The scaling on IaaS level can be illustrated with an example of Amazon Elastic Compute Cloud(AmazonEC2), whereas scaling on PaaS level can be illustrated with Amazon Web Services Elastic Beanstalk(AWS Elastic Beanstalk). AWS Elastic Beanstalk is one of the many services provided by the AWS with its functionality to manage the Infrastructure.Elastic beanstalk provides a quick deployment and management of the applications on the Cloud as it automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. Elastic beanstalk uses highly reliable and scalable services [2].

NEED FOR AWS ELASTIC BEANSTALK

Cloud Computing shifts the location of resources to the cloud to reduce the cost associated with over-provisioning, under-provisioning, and under-utilization. When more than required resources are available, is called over-provisioning [3]. When the resources are not used adequately, it is known as under-utilization. And when the resources available are not enough is called under-provisioning. Cloud Computing should also reduce the time required to provision the resources with the variable workload on the server so that the applications can be quickly scaled up and down with the variable workload [4]. There scaling of applications can be achieved by:

Manual Scaling in Cloud Enviornment

In traditional applications, scalability is achieved by predicting the peak loads, then purchasing, setting up and configuring the infrastructure that could handle this peak load [5]. With Manual Scaling, the resources are provisioned at the deployment time and the application servers are added to infrastructure manually, thus there is high latency. Due to these issues, the average time the applications are provisioned is long. There is also an issue of Manual Monitoring of the resources allocated.

Semi-Manual Scaling in Cloud Enviornment

The resources provided for the Inftrastruare are virtualized in the cloud environment and thus this virtualization enables the elasticity. In particular, in cloud environments, resources are provisioned dynamically (i.e. at runtime), automatically (i.e. without user intervention), infinitely and almost immediately (i.e. within minutes and not hours, days, weeks or months like in traditional environments) [1]. Thus, the mean time until when the resources are provisioned are short. However, manual monitoring of resources is still necessary. In particular, users are forced to make a tradeoff between requesting more resources to avoid under-provisioning and requesting fewer resources to

avoid over-provisioning and under-utilization. Since resources are provisioned by request, the problem of the unavailability of an application at peak loads is not completely eliminated.

Automatic Scaling in Cloud Enviornment

Automatic Scaling enables users to closely follow the workload curve of their application, by provisioning resources on demand. The user owns the choice that the number of resources these applications are using increases automatically during the time when the demand of resources are high to handle the peak load and also automatically decreases when the minimal resources are needed, to minimize the cost so that the user only pays for what they used. Automatic Scaling also predicts the peak load that the applications may require in the future and provisions these required resources in advance, proving the elasticity of the cloud [1].

ELASTIC BEANSTALK SERVICES

Elastic Beanstalk supports applications developed in Java, PHP, NET, Node.js, Python, and Ruby, and also in different container types for each language. A container defines the infrastructure and software stack to be used for a given environment. When an application is deployed, Elastic Beanstalk provisions one or moreAWS resources, such as Amazon EC2 Instances [4]. The software stack that runs on the instances depends on the container type, where two container types are supported by the Elastic Beanstalk Node.js: a 32-bit Amazon Linux Image and a 64-bit Amazon Linux Image. Where each of them runs the Software stack tailored to the hosted Node.js application. Amazon Elastic Beanstalk can be interacted using the AWS Management Console, the AWS Command Line Interface(AWS CLI), or a high-level CLI designed for Elastic BeanStalk [4].

Amazon Elastic Beanstalk provides the Automatic Scaling in cloud Environments. For Automatic Scaling, it uses the following Amazon Web Services:

AWS Management Console

AWS Management Console is a browser-based graphical user interface(GUI) for Amazon Web Services. It allows users to configure an automatic scaling mechanism of AWS Elastic Beanstalk as well as other services of AWS. From the Management console, the user can decide about how many instances does the application require. When the application must be scaled up and down [1].

Elastic Load Balancing

Elastic Load Balancing enables the load balancer which automatically distributes the incoming application traffic across all running instances in the auto-scaling group based on metrics like request count and latency tracked by Amazon Cloud Watch. If an instance is terminated, the load balancer will not route requests to this instance anymore. Rather, it will distribute the requests across the remaining instances.

Elastic Load Balancing also monitors the availability of an application, by checking its "health" periodically (e.g. every five minutes). If this check fails, AWS Elastic Beanstalk will execute further tests to detect the cause of the failure [6]. In particular, it checks if the load balancer and the auto-scaling group are existing. In addition, it checks if at least one instance is running in the auto-scaling group. Depending on the test results, AWS Elastic Beanstalk changes the health status of the application.

The different colors respond to the status of the application. Green indicates that the application responded within a minute, Yellow indicates that the application has not responded in the last 5 minutes, Red indicates that application has not responded in the last 5 minutes or some other problem may have been detected by AWS Elastic Beanstalk. Gray indicates that the status of the application is unknown as of now [6].

Auto Scaling

Auto Scaling automatically launches and terminates instances based on metrics like CPU and RAM utilization of the application and are tracked by Amazon CloudWatch and thresholds called triggers. Whenever a metric crosses a threshold, a trigger is fired to initiate automatic scaling. For example, a new instance will be launched and registered at the load balancer if the average CPU utilization of all running instances exceeds an upper threshold

Auto Scaling also provides fault tolerance. If an instance reaches an unhealthy status or terminates unexpectedly, Auto Scaling will compensate this and launch a new instance instead, thus assuring that the specified minimum number of instances are running constantly.

Amazon Cloud Watch

Amazon CloudWatch is a component of Amazon Web Services (AWS) that provides monitoring for AWS resources and the customer applications running on the Amazon infrastructure [7] It tracks and stores per-instance metrics, including request count and latency, CPU and RAM utilization. Once stored, the metrics can be visualized an API, command-line tools, one of the AWS SDK (software development kits) or the AWS Management Console. With AWS Management Console, users can get real-time visibility into the utilization of each of the instances in an auto-scaling group via the graphs and chart provided, and can easily and quickly detect the over-provisioning, under-utilization or under-provisioning.

CONCLUSION

In traditional (i.e. non-cloud) environments, overprovisioning and under-utilization can hardly be avoided [book-cloudcomuting]. There is an observation that in many companies the average utilization of application servers ranges from 5 to 20 percent, meaning that many resources like CPU and RAM are idle at peak times [8]. On the other hand, if the companies shrink their infrastructures to reduce over-provisioning and under-utilization, the risk of under-provisioning will increase. While the costs of over-provisioning and under-utilization can easily be calculated, the costs of under-provisioning are more difficult to calculate because under-provisioning can lead to a no peak times [8].

Other platforms like Google App Engine [9] also let users have their applications to automatically scale both up and down according to demand but with even more restrictions on how users should develop their applications. Wheras, with AWS Elastic Beanstalk there is more control with the user. A downside of AWS Elastic Beanstalk is that currently, it does not provide any web service to predict demand for the near future. Theoretically, the statistical usage data of the last few months or years could be used to predict time intervals during which more or fewer resources are needed.

REFERENCES

[1] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pfänder, C. Serowy, I. Astrova, S. G. Grivas, and M. Schaaf, "Scaling in cloud environments," *ACM Digital Library*, pp. 145–150, 2011. [Online]. Available: http://www.wseas.us/e-library/conferences/2011/Corfu/COMPUTERS/COMPUTERS-23.pdf

[2] "Waht is aws elastic beanstalk," Web page, Amazon Web Services, 2017, online; accessed 19-Mar-2017. [Online]. Available: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

[3] S. Tai, J. Nimis, C. Baun, and M. Kunze, *Cloud Computing: Web-Based Dynamic IT Services 1st*. Springer Publishing Company, Incorporated ©2011, p. 109. [Online]. Available: http://www.springer.com/us/book/9783642209161

[4] J. Vlie, F. Paganelli, S. van Wel, and D. Dowd, *Elastic Beanstalk*, 1st ed. O'Reilly Media, Inc., 2011, online; accessed 18-Mar-2017.

[5] J. Yang, J. Qiu, and Y. Li, "A profile-based approach to just-in-time scalability for cloud applications." IEEE Computer Society Washington, DC, USA ©2009, 2009, pp. 9–16.

[6] "Elastic load balancing," Web page, Amazon Web Services, online; accessed 20-Mar-2017. [Online]. Available: https://aws.amazon.com/elasticloadbalancing/

[7] "Searchaws CloudWatch," Web page, online; accessed 21-Mar-2017. [Online]. Available: http://searchaws.techtarget.com/definition/CloudWatch

[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Communications of the acm," *ACM Digital Library*, vol. 53, pp. 50–58. [Online]. Available: http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext

[9] "Google cloud platformGoogle App Engine," Web page, online; accessed 18-Mar-2017. [Online]. Available: https://cloud.google.com/appengine/

ASKALON

ABHISHEK NAIK[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: ahnaik@indiana.edu*

project-002, March 26, 2017

ASKALON is a Grid application development and computing environment which aims to provide a Grid to the application developers in an invisible format [1]. This will simplify the development and execution of various workflow applications on the Grid. This will not only allow a transparent Grid access but also will allow the high-level composition of workflow applications. ASKALON basically makes use of five services: Resource Manager, Scheduler, Execution Engine, Performance Analysis and Performance Prediction [2].

Keywords: Cloud, I524

https://github.com/absnaik810/sp17-i524/blob/master/paper2/S17-IR-2022/report.pdf

INTRODUCTION

A Grid is basically a combination of hardware and software that provides reliable and pervasive computation abilities. Grid based development methodologies are used for application development. These Grid-based development methodologies emphasize providing the application developer with a non-transparent grid. ASKALON provides such invisible grid to the application developers. While using ASKALON, the Grid work flow applications are made using Unified Modeling Language (UML) based services [1]. Besides this, it also enables integration of the workflows that have been created programmatically using languages such as XML. ASKALON typically requires some middleware for this.

DESCRIPTION

When using ASKALON, the users need to create Grid workflow applications at a higher level of abstraction using the Abstract Grid Workflow Language (AGWL) which is based on XML. This representation of the workflow is later passed on to the middleware layer for scheduling and exection. Figure 1 shows the typical architecture of ASKALON.

As shown in the figure, the major service components are the Resource Manager which is used for management of the various resources, the Scheduler that is used for scheduling the various workflow applications onto the Grid, the Execution engine which provides reliable and fault tolerant execution, the Performance analyser that analyses the performance and detects bottlenecks and the Performance predictor that estimates the execution times. The following paragraphs provide a brief description about these services:

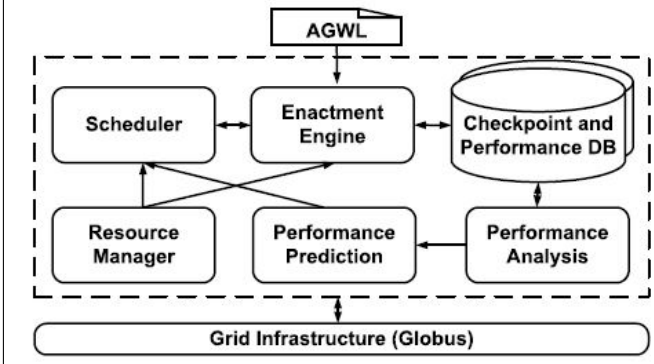


Fig. 1. ASKALON Architecture [1]

Resource Manager

The Resource Manager service is responsible for the management of the resources like the procurement, allocation, reservation and automatic deployment. It usually works hand in hand with the AGWL. In addition to this management of resources, the main task of the Resource manager is to abstract the users from low-level Grid middleware technology.

Scheduler

The Scheduler service is responsible for the scheduling (or mapping) of various workflow applications onto the Grid using optimization algorithms and heuristics based on graphs. In addition to this, it monitors the dynamism of the Grid infrastructures thorough an execution contract and adjusts the optimised static

schedules accordingly. The Scheduler thus provides Quality of Service (QoS) dynamically. It usually uses one out of the three algorithms - Heterogeneous Earliest Finish Time (HEFT), Genetic or Myopic. HEFT algorithm schedules the various workflows by creating an ordered lists of tasks and then mapping the tasks onto the resources in the most efficient way. Genetic algorithms are inspired by Darwin's theory of evolution and work using its the princples of evolution while the Myopic algorithms are basically just-in-time approaches, since they make greedy decisions that would be best in the current scenario.

Execution Engine

The Execution Engine service also known as the Enactment Engine service performs tasks such as checkpointing, retry, restart, migration and replication. It thus aims to provide reliable and fault tolerant execution of the workflows.

Performance Analyser

The Performance Analyser service provides support to the automatic instrumentation and bottleneck detection within the Grid workflow executions. For example, excessive synchronization, load imbalance or non-scalability of the resources might result in a bottleneck and it is the responsibility of the Performance Analyser to detect and report it.

Performance Prediction

The Performance Prediction services predicts the performance, i.e., it emphasises on the execution time of the workflow activities. It uses the training phase and statistical methods for this.

WORKFLOW GENERATION

ASKALON basically offers two interfaces: graphical model based on UML and a programmatic XML based model [1]. The main aim of both these interfaces is to generate large-scale workflows in a compact as well as intuitive form:

UML-modeling based

ASKALON allows creation of workflows via a modeling service similar to UML diagrams. This service combines Activity Diagrams and works in a hierarchical fashion. This service can be implemented in a platform independent way using the Model-View Controller (MVC) paradigm. This service can then be shown to contain three parts: a Graphical User Interface (GUI), model traverser and model checker. This GUI in turn comprises of the menu, toolbar, drawing space, model tree and the properties of the elements. The drawing space can contain several diagrams. The model traverser, as the name suggests, provides a way to move throughout the model visiting each element and accessing its properties. The model checker on the other hand is responsible for the correctness of the model.

XML-based Abstract Grid Workflow Language

The Abstract Grid Workflow Language enables the combination of various atomic units of work called as activities. These activities are interconnected through control-flow and data-flow dependencies. Activities can in turn be of two levels: activity types and activity deployments. An activity type describes the semantics or functions of an activity, whereas the activity deployment points to a deployed Web Service or executable. AGWL is not bounded to any implementation technologies such as the Web Services. Also, the AGWL representation of a typical workflow can be generated in two ways: automatically via the UML representation or manually by the user. In both the cases, AGWL serves as the input to the ASKALON runtime middleware services.

COMPARISON WITH OTHER COUNTERPARTS

Few experiments have been carried out using the seven Grid clusters of the Austrian Grid [3] and a group of 116 CPUs. Figure 2 represents performance of the individual clusters, wherein each cluster aggregates the execution time of all the workflows executed on a single CPU. As can be inferred from the figure, the fastest cluster is around thrice faster than the slowest one.

DAGMan [4] is a scheduler that is used for Condor jobs that have been organized in the form of a Directed Acyclic Graph (DAG). Scheduling doesn't use any specialized optimization techniques and is done simply using matchmaking. Fault tolerance is done using rescue DAG that is automatically generated whenever some activity fails. As against this, the ASKALON checkpointing also takes care of the fact when the Execution Engine itself fails. Thus, the checkpointing provided by ASKALON is more robust compared to that by other counterparts like DAGMan.

ASKALON differs in many respects compared to other projects like Gridbus [5] and UNICORE [6]. In ASKALON, the AGWL allows a scalable specification of many parallel activities by using compact parallel loops. The Enactment Engine also enables handling of very large data collections that are generated by large-scale controls and data-flows. The HEFT and genetic search algorithms that the ASKALON scheduler implements, are not used by the other projects, like the ones mentioned above. The Enactment Engine also provides checkpointing of workflows at two leves for restoring and resuming, in case the engine itself fails. In ASKALON, a lot of emphasis is laid on the clear separation between the Scheduler and the Resource Manager. It thus proposes a novel architecture in terms of physical and logical resources and thus provides brokerage, reservations and activity type to deployment mappings.

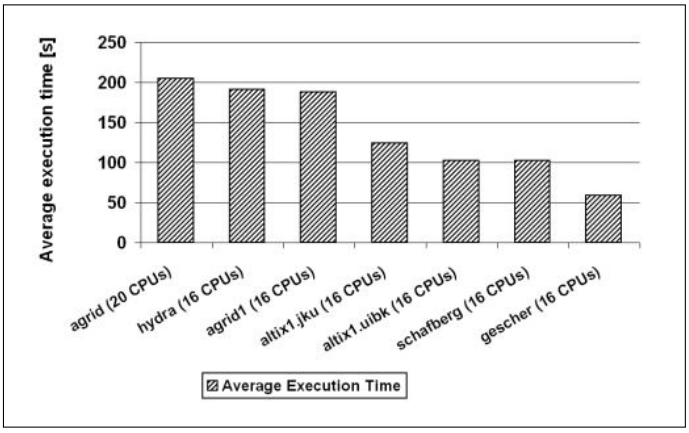


Fig. 2. Performance of Austrian Grid clusters [3]

ASKALON IN BIG DATA PROJECTS

ASKALON has been used in the design of Meteorological Stimulations in the Cloud [7]. In order to deploy the application on a distributed Grid infrastructure, the simulation code was split

into a workflow called the RainCloud, which was represented using ASKALON. Figure 3 denotes the graphical representation of this workflow.

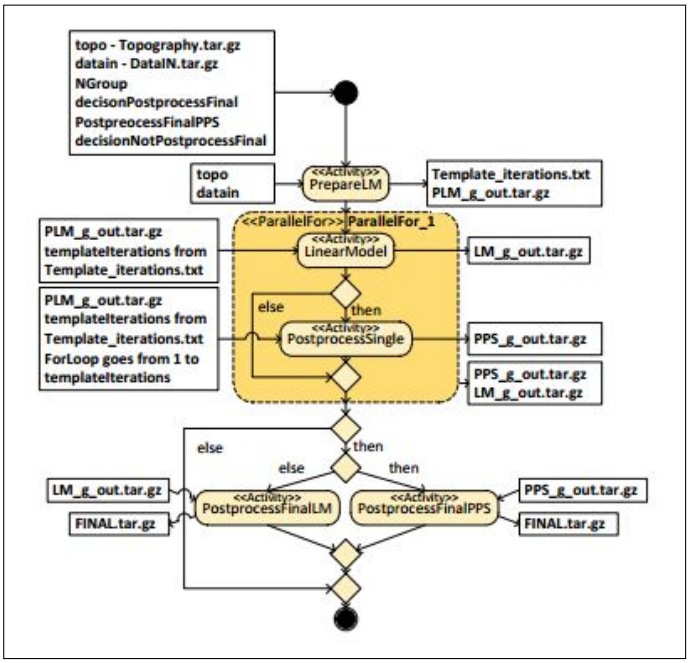


Fig. 3. Graphical representation of the Meteorological workflow [7]

This workflow was flexible in the sense that it could be easy extended to suite the needs of some other projects. For example, this workflow setup was extended for an investigation of precipitation/snow accumulation on the Kongsvegen glacier on Svalbard, Norway, as a part of the SvalClac project [7].

CONCLUSION

ASKALON supports workflow integration using UML and also provides an XML based programming interface. This effectively abstracts the end user from the low level middleware technologies. The Resource Manager handles the logical and physical resources and the workflow activities to provide features such as authorization, Grid resource discovery, selection, allocation and interaction. Scheduler makes use of some algorithms like HEFT or other genetic algorithms which deliver high performance. It highly benefits from the Performance Prediction service which in turn depend upon the training phase and the statistical methods used. The Execution Engine handles data dependencies and also working on high volume data - something that is highly useful in Big Data related applications and projects. The Performance Analyser analysis the performance benchmarks. Typically, the overhead of ASKALON middleware services which consist of the Resource Manager, Scheduler and Performance prediction are usually constant, thereby requiring less execution time holistically.

Thus, to conclude, we focused on ASKALON as a Grid application development environment. We also saw the various architectural components of ASKALON as well as the comparisons amongst the different Grid clusters that were used. We also saw some Big Data use cases wherein ASKALON was used and the flexibility with which the workflow setup using ASKALON was extended to support different projects.

REFERENCES

[1] I. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer, 2006, pp. 462–471.

[2] F. Pop, J. Kolodziej, and B. DiMartino, Eds., *Resource Management for Big Data Platforms*. Springer Nature, 2016, pp. 49–50.

[3] "Austrian-grid-project," Web Page, accessed: 2017-3-20. [Online]. Available: http://austriangrid.at

[4] "Dagman-manager," Web Page, accessed: 2017-3-19. [Online]. Available: http://www.cs.wisc.edu/condor/dagman

[5] R. Buyya and S. Venugopal, "Gridbus technologies," in *The Gridbus toolkit for Service Oriented Grid and Utility Computing*, 2004. [Online]. Available: http://www.cloudbus.org/papers/gridbus2004.pdf

[6] "Unicore technologies," Web Page, accessed: 2017-3-20. [Online]. Available: http://www.springer.com/gp/computer-science/lncs

[7] G. Morar, F. Schuller, S. Ostermann, R. Prodan, and G. Mayr, "Meteorological simulations in the cloud with the askalon environment," in *Euro-Par 2012 Parallel Processing Workshops*, vol. 7640, 2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-36949-0_9?CFID=916124568&CFTOKEN=29254584

Memcached

RONAK PAREKH[1] AND GREGOR VON LASZEWSKI[2]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

project-000, March 26, 2017

Memcached is a free and open source, high-performance, distributed memory object caching system. It allows the system to make better use of its memory. It uses data caching technique to speed up dynamic database-driven websites. This reduces the number of times an external data source is accessed by the application. Memcached service is mainly offered through an API. It allows to take memory from parts of the system where you have more memory and allocate it to those parts of the system where you have less memory.

Keywords: Cloud, I524

https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf

INTRODUCTION

Memcached is a high-performance, distributed memory object caching system, generic in nature, but originally intended for use in speeding up dynamic web applications by alleviating database load [1]. Memcached allows you to take memory from parts of your system where you have more than you need and make it accessible to areas where you have less than you need. It is a free and open-source software, licensed under the Revised BSD [2] license. Memcached runs on Unix-like operating systems (at least Linux and OS X) and on Microsoft Windows. It depends on the libevent library.

COMPARISON OF MEMCACHED AND REDIS

- Server: Memcached server is multi-threaded whereas Redis is single threaded. As a result, Redis [3] can't effectively harness multiple cores.
- Distributability: Memcached is very easy to distribute since it doesn't support any rich data-structures. Redis, on the other hand, is much harder to distribute without changing semantics/performance guarantees of some its commands.
- Ease of Installation: Comparing ease of Installation Redis is much easier. No dependencies required.
- Memory Usage: For simple key-value pairs memcached is more memory efficient than Redis. If you use Redis hashes, then Redis is more memory efficient.
- Persistence: If you are using Memcached then data is lost with a restart and rebuilding cache is a costly process. On the other hand, Redis can handle persistent data. By default, Redis syncs data to the disk at least every 2 seconds.
- Replication: Memcached does not supports replication. Whereas Redis supports master-slave replication. It allows slave Redis servers to be exact copies of master servers.Data from any Redis server can replicate to any number of slaves [4].
- Read/Write Speed: Memcached is very good to handle high traffic websites. It can read lots of information at a time and give you back at a great response time. Redis can also handle high traffic on read but also can handle heavy writes as well.
- Key Length: Memcached key length has a maximum of 250 bytes, whereas Redis key length has a maximum of 2GB. When advanced data structures or disk-backed persistence are important, Redis is used.

ARCHITECTURE OF MEMCACHED

Memcached implements a simple and lightweight key-value interface where all key-value tuples are stored in and served from DRAM. The following commands are used by clients to communicate with the Memcached servers:

- SET/ADD/REPLACE(key, value): add a (key, value) object to the cache.
- GET(key): retrieve the value associated with a key.
- DELETE(key): delete a key

Internally, a hash table is used to index the key-value entries. Memcached uses a hash table to index the key-value entries. These entries are also in a linked list sorted by their most recent access time. The least recently used (LRU) entry is evicted and replaced by a newly inserted entry when the cache is full [5].

Hash Table: To lookup keys quickly, the location of each key-value entry is stored in a hash table. Hash collisions are resolved by chaining: if more than one key maps into the same hash table bucket, they form a linked list. Chaining is efficient for inserting or deleting single keys. However, lookup may require scanning the entire chain.

Memory Allocation: Naive memory allocation could result in significant memory fragmentation. To address this problem, Memcached uses slab-based memory allocation. Memory is divided into 1 MB pages, and each page is further sub-divided into fixed-length chunks. Key-value objects are stored in an appropriately sized chunk. The size of a chunk, and thus the number of chunks per page, depends on the particular slab class. For example, by default the chunk size of slab class 1 is 72 bytes and each page of this class has 14563 chunks; while the chunk size of slab class 43 is 1 MB and thus there is only 1 chunk spanning the whole page. To insert a new key, Memcached looks up the slab class whose chunk size best fits this key-value object [5]. If a vacant chunk is available, it is assigned to this item; if the search fails, Memcached will execute cache eviction.

Cache policy In Memcached, each slab class maintains its own objects in an LRU queue (see Figure 1). Each access to an object causes that object to move to the head of the queue. Thus, when Memcached needs to evict an object from the cache, it can find the least recently used object at the tail. The queue is implemented as a doubly-linked list, so each object has two pointers.

Threading: Memcached was originally single-threaded. It uses libevent for asynchronous network I/O callbacks. Later versions support multi-threading but use global locks to protect the core data structures. As a result, operations such as index lookup/update and cache eviction/update are all serialized [5].

WORKING OF MEMCACHED

In the Memcached system, each item comprises a key, an expiration time, optional flags, and raw data. When an item is requested, Memcached checks the expiration time to see if the item is still valid before returning it to the client. The cache can be seamlessly integrated with the application by ensuring that the cache is updated at the same time as the database. By default, Memcached acts as a Least Recently Used cache plus expiration timeouts. If the server runs out of memory, it looks for expired items to replace. If additional memory is needed after replacing all the expired items, Memcached replaces items that have not been requested for a certain length of time (the expiration timeout period or longer), keeping more recently requested information in memory.

Working of Memcached Client [6]: Firstly, a memcached client object is created and it starts calling its method to get and set cache entries. When an object is added to the cache, the Memcached client will take that object, serialize it, and send a byte array to the Memcached server for storage. At that point, the cached object might be garbage collected from the JVM where the application is running. When the cached object is needed, the Memcached client's get() method is called. The client will take the get request, serialize it, and send it to the Memcached server. The Memcached server will use the request to look up the object from the cache. Once it has the object, it will return the byte array back to the Memcache client. The client object will then take the byte array and deserialize it to create the object and return it to the application. Even if the application runs on more than one server, all of them can point to the same Memcached server and use it for getting and setting cache entries. The Memcached client is configured in such a way that it knows all the available Memcached servers.

SECURITY

Memcached is mostly used for deployments within a trusted network. However, sometimes Memcached is deployed in untrusted networks and enviroments where administrators exercise control over the clients that are connected. SASL authentication support is required for Memcached to compile and it requires the binary protocol. For simplicity, all Memcached operations are treated equally. Clients with a valid need for access to low-security entries within the cache gain access to all entries within the cache. If the cache key can be either predicted, guessed or found by exhaustive searching, its cache entry may be retrieved [7].

NEW FEATURES

Chained items were introduced in 1.4.29. With -o modern items sized 512k or higher are created by chaining 512k chunks together. This made increasing the max item size (-I) more efficient in many scenarios as the slab classes no longer have to be stretched to cover the full space. There was still an efficiency hole for items 512k->5mb or so where the overhead is too big for the size of the items. This change fixes it by using chunks from other slab classes in order to "cap" off chained items. With this change larger items should be more efficient than the original slab allocator in all cases.Chunked items are only used with -o modern or explicitly changing -o slab-chunk-max It is not recommended to set slab-chunk-max to be smaller than 256k at this time.

USES OF MEMCACHED

Memcached is used for scaling large websites. Facebook is one of the largest users of memcached [8]. It is used to alleviate database load. Facebook uses more than 800 servers supplying over 28 terabytes of memory to our users. As Facebook's popularity had skyrocketed, they've run into a number of scaling issues. This ever increasing demand required Facebook to make modifications to both our operating system and memcached to achieve the performance that provided the best possible experience for their users. There were thousand of computers, each running hundreds of Apache processes which ultimately led to thousands of TCP connections open to the memcached processes. Memcached uses a per-connection buffer to read and write data out over the network. When hundreds and thousands of connections were in consideration, memory added up to be in gigabytes. Memory that could be better used to store user data had to be considered. Thus, memcached came into consideration while scaling large websites.

CONCLUSION

Memcached is used when scaling large websites is to be done. By alleviating the database load, it helps making the maximum use of its caching technique to find hits before the database is queried. This results in lesser amount of cycles to and from the external data source or database when it comes to speeding up query results. Memcached has its main advantage in distributability when it comes to setting up a distributed cache and gives superior performance when multithreaded processes are in consideration.

REFERENCES

[1] Dormando, “What is memcached?” Web Page, Mar. 2015, accessed 2017-03-21. [Online]. Available: https://memcached.org/

[2] Wikipedia, “Berkeley software distribution,” Web Page, Mar. 2017, accessed 2017-03-22. [Online]. Available: https://en.wikipedia.org/wiki/Berkeley_Software_Distribution

[3] Wikipedia, “Redis,” Web Page, Mar. 2017, accessed 2017-03-22. [Online]. Available: https://en.wikipedia.org/wiki/Redis

[4] Squarespace Inc., “Memcached vs redis,” Web Page, Mar. 2014, accessed 2017-03-25. [Online]. Available: http://www.openldap.org/lists/openldap-software/200010/msg00097.html

[5] S. M., “Memcached vs redis,” Blog, Mar. 2014, accessed: 23-Mar-2017. [Online]. Available: http://blog.andolasoft.com/2014/02/memcached-vs-redis-which-one-to-pick-for-large-web-app.html

[6] Sunil Patil, “Use memcached for java enterprise performance,” Web Page, Mar. 2012, accessed 2017-03-24. [Online]. Available: http://www.javaworld.com/article/2078565/open-source-tools/open-source-tools-use-memcached-for-java-enterprise-performance-part-1-architecture-and-setup.html

[7] Wikipedia, “Memcached,” Web Page, Feb. 2017, online; accessed 23-Mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Memcached

[8] Facebook, “Scaling memcached at facebook,” Web Page, Mar. 2015, accessed 2017-03-22. [Online]. Available: https://www.facebook.com/note.php?note_id=39391378919&ref=mf

Naiad

RAHUL RAGHATATE[1,*] AND SNEHAL CHEMBURKAR[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: rragahta@iu.edu, snehchem@iu.edu

paper-2, March 26, 2017

Naiad is a distributed system based on computational model called timely dataflow developed for execution of data-parallel, cyclic dataflow programs. It provides an in-memory distributed dataflow framework which exposes control over data partitioning and enables features like the high throughput of batch processors, the low latency of stream processors, and the ability to perform iterative and incremental computations [1]. These features allow the efficient implementation of many dataflow patterns, from bulk and streaming computation to iterative graph processing and machine learning. This paper explaines the Naiad Framework, its abstractions, and the reasoning behind it.

Keywords: Naiad, iterative, dataflow, graph, stream

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2026/report.pdf

INTRODUCTION

Abundance of distributed dataflow systems around the big data ecosystem has resulted in achieving many goals like high throughput, incremental updating, low latency stream processing, iterative computation. Many graph analytics procedures involves incremental as well as iterative processing. However, most data-parallel systems support either incremental update or iteration, but not both.

Frameworks like descendants of MapReduce [2, 3],stream processing systems [4, 5], materialized view-maintenance engines [6], provide efficient support for incremental input, but do not support iterative processing. On the other hand, iterative data-parallel frameworks like Datalog [7], recursive SQL databases [8] and systems adding iteration over MapReduce like model [9–12] provide a constrained programming model (e.g. stratified negation in Datalog) and none supports incremental input processing [13].

Map Reduce, being functional language,requires passing the entire state of the graph from one stage to the next, which is inefficient. In real-time applications batch processing delays of MapReduce may prove unacceptable [14].

With a goal to find common low-level abstraction and system design that could be re-used for all of these computational workloads and to reduce the engineering cost of domain-specific distributed systems by allowing them to share a single highly optimized core codebase, Naiad, a timely dataflow system was developed at Microsoft by Derek G. Murray *et al.* [15].

Naiad extends standard batch data-parallel processing models like MapReduce, Hadoop, and Dryad/DryadLINQ, to support efficient incremental updates to the inputs in the manner of a stream processing system, while at the same time enabling arbitrarily nested fixed-point iteration [1].

Naiad, a distributed system for executing data parallel cyclic dataflow programs offers high throughput batch processing, low latency stream processing and the ability to perform iterative and incremental computations all in one framework [16].

In Naiad, the underlying computation framework provides a mechanism for iteration and incremental update, while the language provides a structured means of expressing these computations. This yields both an expressive programming model and an efficient implementation [13].

ARCHITECTURE

The Naiad architecture consists of two main components- (1) incremental processing of incoming updates and (2) low-latency real-time querying of the application state.

Many data processing tasks require low-latency interactive access to results, iterative sub-computations, and consistent intermediate outputs so that sub-computations can be nested and composed.Figure 1 shows a Naiad application that supports realtime queries on continually updated data. The dashed rectangle represents iterative processing that incrementally updates as new data arrive [1].

From Figure 1, it can be seen that query path is clearly separated from the update path. This results in query processing separately on a slightly stale version of the current application state and the query path does not get blocked or incur delays due to the update processing. This also resolves complex situations: If queries shared the same path with updates, the queries could be accessing partially processed/incomplete/inconsistent states, which would have to be taken care out separately.

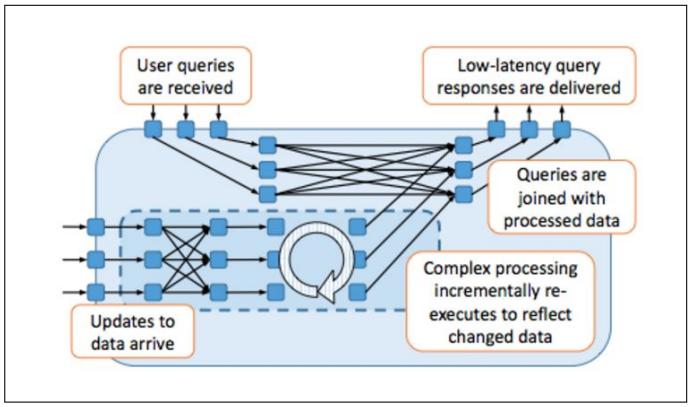


Fig. 1. Naiad Application that support real time querries on continually updated data [1].

Even though hybrid approach to assemble the application in Figure 1 by combining multiple existing systems have been widely deployed, applications built on a single platform as in Figure 1 are typically more efficient, succinct, and maintainable [1].

Features

Naiad being a timely dataflow computational model, enriches dataflow computation with timestamps that represent logical points in the computation and provide the basis for an efficient, lightweight coordination mechanism.Timely dataflow supports the following three features:

1. Structured loops allowing feedback in the dataflow
2. Stateful dataflow vertices capable of consuming and producing records without global coordination, and
3. Notifications for vertices once they have received all records for a given round of input or loop iteration.

Structured loops and Stateful dataflow vertices allows iterative and incremental computations with low latency.Notifications makes Naiad possible to produce consistent results, at both outputs and intermediate stages of computations, in the presence of streaming or iteration [1].

TIMELY DATAFLOW

Applications should produce consistent results, and consistency requires coordination, both across dataflow nodes and around loops [1]. Timely dataflow is a computational model that attaches virtual timestamps to events in structured cyclic dataflow graphs providing coordination mechanism that allows low-latency asynchronous message processing while efficiently tracking global progress and synchronizing only where necessary to enforce consistency. Naiad model simply checkpoints its state periodically, restoring the entire system state to the most recent checkpoint on failure. Even if its not a sophisticated design,it was chosen in part for its low overhead. Faster common-case processing allows more computation to take place in the intervals between checkpointing, and thus often decreases the total time to job completion [15].

Asynchronous messaging

All dataflow models require some communication means for message passing between node over outgoing edges. In a timely dataflow system, each node implements an *OnRecv* event handler that the system can call when a message arrives on an incoming edge, and the system provides a *Sendmethod* that a node can invoke from any of its event handlers to send a message on an outgoing edge [15]. Messages are delivered asynchronously.

Consistency

Computations like reduction functions *Count* or *Average* include subroutines that must accumulate all of their input before generating an output. At the same time, distributed applications commonly split input into small asynchronous messages to reduce latency and buffering. For timely dataflow to support incremental computations on unbounded streams of input as well as iteration, it has a mechanism to signal when a node (or data-parallel set of nodes) has seen a consistent subset of the input for which to produce a result [15].

Iterative Graph Dataflow

A Naiad dataflow graph is acyclic apart from structurally nested cycles that correspond to loops in the program. The logical timestamp associated with each event represents the batch of input that the event is associated with, and each subsequent integer gives the iteration count of any (nested) loops that contain the node. Every path around a cycle includes a special node that increments the innermost coordinate of the timestamp. The system enforced rule restricts event handler from sending a message with a time earlier than the timestamp for the event it is handling ensuring a *partial order* on all of the pending events (undelivered messages and notifications) in the system,thus enabling efficient progress tracking [1].

Progress Tracking

The ability to deliver notifications(an event that fires when all messages at or before a particular logical timestamp have been delivered to a particular node) promptly and safely is critical to a timely dataflow system's ability to support low-latency incremental and iterative computation with consistent results.Naiad can implements progress trackers to establish the guarantee that no more messages with a particular timestamp can be sent to a node [1].

Achieving Timely dataflow

Each event has a timestamp and a location (either a vertex or edge), and are referred as pointstamp. The timely dataflow graphs structure ensures that, for any locations l_1 and l_2 connected by two paths with different summaries, one of the path summaries always yields adjusted timestamps earlier than the other. For each pair l_1 and l_2, we find the minimal path summary over all paths from l_1 to l_2 using a straightforward graph propagation algorithm, and record it as $\Psi[l_1, l_2]$. To efficiently evaluate the couldresult-in relation for two pointstamps (t_1, l_1) and (t_2, l_2), we test whether $\Psi[l_1, l_2](t_1) \leq t_2$ [1].

Informally, Timely Dataflow supports directed dataflow graphs with structured cycles, analogous to structured loops in a standard imperative programming language. This structure provides information about where records might possibly flow in the computation, allowing an implementation like Naiad to efficiently track and inform dataflow vertexes about the possibility of additional records arriving at given streaming epochs or iterations [17].

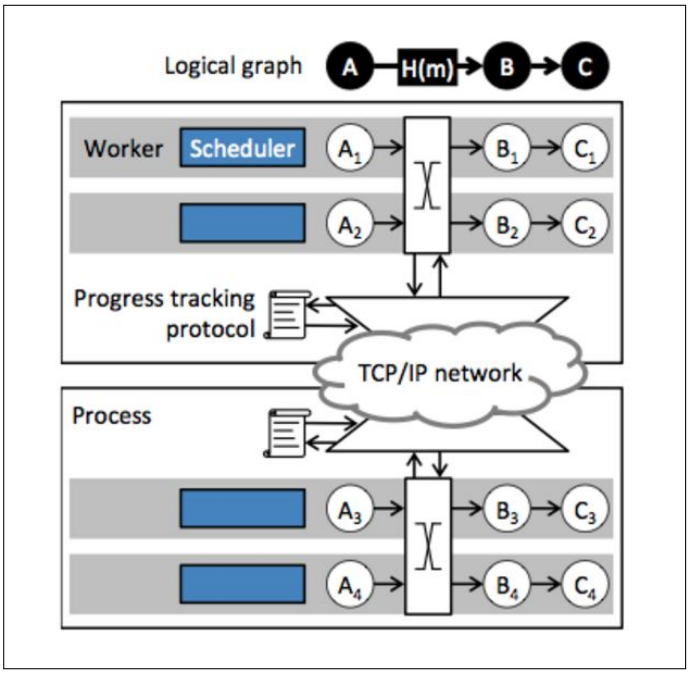


Fig. 2. The mapping of a logical dataflow graph onto the distributed Naiad system architecture [1].

SYSTEM IMPLEMENTATION

Naiad is high-performance distributed implementation of timely dataflow and is written in C#, and runs on Windows, Linux, and Mac OS. Figure 2 shows the schematic architecture of a Naiad cluster consisting a group of processes hosting workers that manage a partition of the timely dataflow vertices [15]. Each worker may host several stages of the dataflow graph. The workers are data nodes and keep a portion of the input data (usually a large-scale input graph, such as Twitter follower graph) in memory. So it makes sense to move computation (dataflow graph stages) to the data (the partitioned input graph)[14]. Each process participates in a distributed progress tracking protocol, in order to coordinate the delivery of notifications. All of the features of C#, including classes, structs, and lambda functions, to build a timely dataflow graph from a system-provided library of generic Stream objects can be implemented. Naiad model uses deferred execution method: like adding a node to internla data flow graph at runtime while executing a method like Max on a Stream [1].

Naiad's distributed implementation also exhibits features like distributed progress tracking, a simple but extensible implementation of fault tolerance, high availability, avoidance of micro-straggler (events like packet loss, contention on concurrent data structures, and garbage collection which leads to delays ranging from tens of milliseconds to tens of seconds) which is main obstacle to scalability for low-latency workloads [1].

PERFORMANCE EVALUATION

Naiad's performance over supporting high-throughput, low latency, data-parallelism, batch processing, iterative graph data processing have been examined using several notable micro-benchmarks by Murray *et al.* [1] such as Throughput, Latency, Protocol Optimizations, Scaling.

USE CASES

Naiad framework has been majorly deployed for batch, streaming and graph computations serving interactive queries against the results where Naiad responds to updates and queries with sub-second latencies [1].

Batch iterative graph computation

Implementation of graph algorithms such as PageRank, strongly connected components (SCC), weakly connected components(WCC), and approximate shortest path(ASP) in Naiad requires less code complexity and provides dominant difference in running times compared to PDW [18], DryadLINQ [19],SHS [20] for Category A web graph [1, 21, 22].

Batch iterative machine learning

Naiad provides competitive platform for custom implementation of distributed machine learning. Also it is straightforward to build communication libraries for existing applications using Naiad's API. For Eg., modified version of Vowpal Wabbit (VW), an open-source distributed machine learning library which performs iterative linear regression [23]. AllReduce (processes jointly performing global averaging) implementation requires 300 lines of code, around half as many as VW's AllReduce, and the Naiad code is at a much higher level, abstracting the network sockets and threads being used [1].

Streaming acyclic computation

Latency reduction while computing the k-exposure metric for identifying controversial topics on Twitter using Kineograph, which takes snapshots of continuosly ingesting graph data for data parallel computations [24].

Streaming iterative graph analytics

Twitter Analysis: To compute the most popular hashtag in each connected component of the graph of users mentioning other users, and provide interactive access to the top hashtag in a user's connected component.

USEFUL RESOURCES

Naiad: A Timely Dataflow System [1], provides extensive study of Naiad's architecture, methodology of working, its distributed implementation, iterative and incremental data processing, data analysis applications, comparison with other graph processing frameworks. Moreover, [25], [26], [13] are also good resources to learn about Naiad and programming in Naiad Framework.

CONCLUSION

Naiad model enriches dataflow computation with timestamps that represent logical points in the computation and provide the basis for an efficient, lightweight coordination mechanism. All the above capabilities in one package allows development of High-level programming models on Naiad which can perform tasks as streaming data analysis, iterative machine learning, and interactive graph mining. Moreover, it's public reusable low-level programming abstractions leads Naiad to outperforms many other data parallel systems that enforce a single high-level programming model.

ACKNOWLEDGEMENTS

This work was done as part of the course "I524: Big Data and Open Source Software Projects" at Indiana University during Spring 2017. Many thanks to Professor Gregor von Laszewski and Prof. Geoffrey Fox at Indiana University Bloomington for their academic as well as professional guidance. We would also like to thank Associate Instructors for their help and support during the course.

REFERENCES

[1] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: ACM, 2013, pp. 439–455, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/2517349.2522738

[2] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 7:1–7:14, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038923

[3] P. K. Gunda, L. Ravindranath, C. A. Thekkath, Y. Yu, and L. Zhuang, "Nectar: Automatic management of data and computation in datacenters," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 75–88, accessed: 2017-3-22. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924949

[4] R. S. Barga, J. Goldstein, M. H. Ali, and M. Hong, "Consistent streaming through time: A vision for event stream processing," *CoRR*, vol. abs/cs/0612115, 2006, accessed: 2017-3-22. [Online]. Available: http://arxiv.org/abs/cs/0612115

[5] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo, "Spade: The system s declarative stream processing engine," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1123–1134, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376729

[6] A. Gupta and I. S. Mumick, "Materialized views," A. Gupta and I. S. Mumick, Eds. Cambridge, MA, USA: MIT Press, 1999, ch. Maintenance of Materialized Views: Problems, Techniques, and Applications, pp. 145–157, accessed: 2017-3-24. [Online]. Available: http://dl.acm.org/citation.cfm?id=310709.310737

[7] S. Ceri, G. Gottlob, and L. Tanca, "What you always wanted to know about datalog (and never dared to ask)," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 1, pp. 146–166, Mar 1989, accessed: 2017-3-24.

[8] A. Eisenberg and J. Melton, "Sql: 1999, formerly known as sql3," *SIGMOD Rec.*, vol. 28, no. 1, pp. 131–138, Mar. 1999, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/309844.310075

[9] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 285–296, Sep. 2010, accessed: 2017-3-22. [Online]. Available: http://dx.doi.org/10.14778/1920841.1920881

[10] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 810–818, accessed: 2017-3-23. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851593

[11] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "Ciel: A universal execution engine for distributed data-flow computing," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 113–126, accessed: 2017-3-24. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972470

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2, accessed: 2017-3-24. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228301

[13] F. McSherry, R. Isaacs, M. Isard, and D. Murray, "Composable incremental and iterative data-parallel computation with naiad," Tech. Rep. MSR-TR-2012-105, October 2012, accessed: 2017-3-24. [Online]. Available: https://www.microsoft.com/en-us/research/publication/composable-incremental-and-iterative-data-parallel-computation-with-naiad/

[14] Edgar, "Metadata and the naiad post – one other good blog on data science/engineering," Blog, Jan. 2017, accessed: 23-Mar-2017. [Online]. Available: https://theinformationageblog.wordpress.com/2017/01/09/metadata-and-the-naiad-post-one-other-good-blog-on-data-scienceengineering/

[15] D. G. Murray, F. McSherry, M. Isard, R. Isaacs, P. Barham, and M. Abadi, "Incremental, iterative data processing with timely dataflow," *Commun. ACM*, vol. 59, no. 10, pp. 75–83, Sep. 2016, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/2983551

[16] HU2LA, "Review of "naiad: A timely dataflow system" < huula . webdesign + ai," Blog, Sep. 2015, accessed: 23-Mar-2017. [Online]. Available: https://huu.la/blog/review-of-naiad-a-timely-dataflow-system

[17] Microsoft, "Naiad-microsoft," Web Page, Microsoft, 2017, accessed: 2017-3-22. [Online]. Available: https://www.microsoft.com/en-us/research/project/naiad/

[18] Microsoft, "Microsoft analytics platform system overview |microsoft," Web Page, Microsoft, 2017, accessed: 2017-3-26. [Online]. Available: https://www.microsoft.com/en-us/sql-server/analytics-platform-system

[19] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "Dryadlinq: A system for general-purpose distributed data-parallel computing using a high-level language," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–14, accessed: 2017-3-24. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855741.1855742

[20] M. Najork, "The scalable hyperlink store," in *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, ser. HT '09. New York, NY, USA: ACM, 2009, pp. 89–98, accessed: 2017-3-23. [Online]. Available: http://doi.acm.org/10.1145/1557914.1557933

[21] Lemur, "The clueweb09 dataset," Web Page, Red Hat, Inc., 2017, accessed: 2017-3-24. [Online]. Available: http://lemurproject.org/clueweb09/

[22] M. Najork, D. Fetterly, A. Halverson, K. Kenthapadi, and S. Gollapudi, "Of hammers and nails: An empirical comparison of three paradigms for processing large graphs," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, ser. WSDM '12. New York, NY, USA: ACM, 2012, pp. 103–112, accessed: 2017-3-24. [Online]. Available: http://doi.acm.org/10.1145/2124295.2124310

[23] John Lanford, "Home . johnlangford/vowpal_wabbit wiki," Code Repository, Dec. 2016, accessed: 2017-3-24. [Online]. Available: https://github.com/JohnLangford/vowpal_wabbit/wiki

[24] R. Cheng, J. Hong, A. Kyrola, Y. Miao, X. Weng, M. Wu, F. Yang, L. Zhou, F. Zhao, and E. Chen, "Kineograph: Taking the pulse of a fast-changing and connected world," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 85–98, accessed: 2017-3-22. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168846

[25] Microsoft Research, "Microsoft/naiad:the naiad system provides fast incremental and iterative computation for data-parallel workloads," Code Repository, Nov. 2014, accessed: 2017-3-24. [Online]. Available: https://github.com/MicrosoftResearch/Naiad

[26] naiadquestions@microsoft.com, "Nuget gallery|naiad - core 0.5.0-beta," Web Page, NET Foundation, Nov. 2014, accessed: 2017-3-26. [Online]. Available: https://www.nuget.org/packages/Microsoft.Research.Naiad/

Dryad : Distributed Execution Engine

SHAHIDHYA RAMACHANDRAN[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: shahrama@iu.edu*

Technology paper 2, March 26, 2017

Dryad is a general purpose execution environment for distributed, data parallel applications and it automatically handles job creation and management, resource management, job monitoring and visualization, fault tolerance, re-execution, scheduling, and accounting. It creates a dataflow graph by using computational 'vertices' and communication 'channels'. Dryad is the middleware abstraction that is independent of the semantics of the program. It is not suitable for real-time processing since it focuses on throughput rather than latency.

Keywords: Dryad, Distribute Processing, Concurrent Processing, Dryad LINQ, Dataflow, Cluster Computing

https://github.com/shah0112/sp17-i524/paper2/S17-IR-2027/report.pdf

1. INTRODUCTION

With the expansion of the large scale Internet services that depend on clusters of thousands of servers it became more complicated to manage wide-area distributed computing. Some of the prominent challenges included high-latency, unreliable networks, control of resources by separate federated or competing entities, and issues of identity for authentication and access control [1]. Dryad was launched in an effort to make it easier for developers to write efficient parallel and distributed applications without having in-depth knowledge of concurrent programming [2] It focused primarily on reliability, efficiency and scalability of the applications. Dryad was launched as a research project at MSR(Microsoft Research) and was released in EuroSys'07, March 21–23, 2007, Lisboa, Portugal [3].
A Dryad application is a graph generator which can synthesize any given directed acyclic graph. These graphs are dynamic and can change depending on computations made during runtime [2]. The vertices of the Directed Acyclic Graph define the operations that are to be performed on the data. These 'computational vertices' are written as single threaded sequential C++ constructs and are connected using one-way channels. Channels facilitate communication between the vertices through temporary files, TCP/IP streams, and shared-memory FIFOs [1]. The graph is parallelized by distributing the vertices across multiple execution engines. Dryad can scale this across multiple processor cores on the same computer or different physical computers connected by a network, as in a cluster [1]. Scheduling of the computational vertices on the available hardware is handled by the Dryad run-time, without any explicit intervention by the developer of the application or administrator of the network.

2. DRYAD SYSTEM

A Dryad job is a directed acyclic graph where each vertex is a program and edges represent data channels. The logical computation graph is automatically mapped onto physical resources by run-time. The number of vertices in the graph can be much greater than the number of execution cores in the computing cluster. During run-time a finite sequence of 'structured items' are transported through the channels. channels produce and consume heap objects that inherit from a base type. The vertex program then reads and writes its data in the same way irrespective of how the channel serializes its data. Each application has its own serialization/deserialization routine [4].

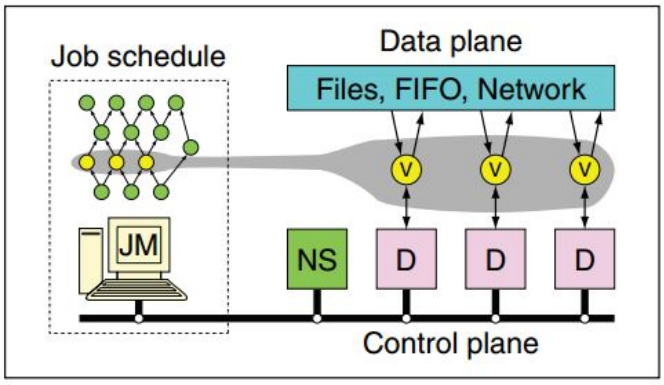


Fig. 1. The Dryad system organization Source:[4]

The overall structure of a Dryad system is shown in the Figure 1. The job manager (JM) consults the name server (NS) to discover

the list of available computers. It maintains the job graph and schedules running vertices (V) as computers become available using the daemon (D) as a proxy. Vertices exchange data through files, TCP pipes, or shared-memory channels. The shaded bar indicates the vertices in the job that are currently running [4].

2.1. Job Manager(JM)

A Dryad job is coordinated by the job manager. It runs either within the cluster or on a user's workstation with network access to the cluster. It contains the code to construct the communication graph and also the library code to schedule the work across the available resources. Job manager handles only the control decisions and is not involved in the actual data transfer between vertices [4]

2.2. Name Server(NS)

The Name Server lists all the available computers and exposes the position of each computer within the network topology so that scheduling decisions can be taken by taking locality into consideration [4]

2.3. Daemon(D)

It is a cluster that runs on each of the computers and is responsible for creating processes on behalf of the job manager. The first time a vertex (V) is executed its binary is sent from the job manager to the daemon. It acts as a proxy to the job manager when it checks the status of the computation and how much data has been read and written on the channels of the remote vertices [4]

2.4. Job Scheduler

The task scheduler is used to queue batch jobs. A distributed storage system is used that allows large files to be broken into small pieces, replicated and distributed across the local disks of the cluster computers [4]

3. DRYAD GRAPH

Each graph is represented as $G =< V_G, E_G, I_G, O_G >$ [4] where V_G is a sequence of vertices with E_G directed edges and two sets $I_G \subset V_G$ and $O_G \subset V_G$ indicate the input and output vertices respectively. Directed Acyclic graphs are used because a vertex can run anywhere once its inputs have been received, it does not lead to a dead-lock situation and the finite length channels ensure that the process will definitely terminate [5]. Figure 2 shows an example of a Directed Acyclic Graph. One of the primary restrictions on a Dryad job is that it has to be representable as a Directed Acyclic Graph. The graph can have multiple edges between the vertices. Each circle represents a program that processes a set of inputs and outputs the results. The inputs and outputs in the graph are virtual vertices of the distributed Dryad system.

3.1. Inputs and Outputs

Large input files are partitioned and distributed across the computers of the cluster. The input is grouped into a graph $G = V_P, \phi, \phi, V_P$ where V_P [6] is a sequence of virtual vertices corresponding to the partitions of the input. After the job is completed, a set of output partitions are concatenated to form a single named distributed file. An application will generally interrogate its input graphs to read the number of partitions at run-time and automatically generate the appropriately replicated graph.

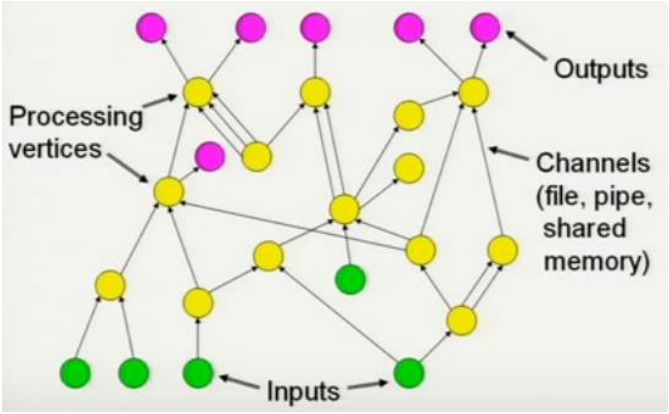


Fig. 2. Directed Acyclic Graph in Dryad job Source:[5]

3.2. Creating new vertices

All vertex programs inherit from the Dryad defined C++ base class. Each program has a textual name - unique within the application and a static factory- that has knowledge of constructing it [6]. A graph vertex is created by calling the appropriate static program factory. Vertex specific parameters are set by calling methods on the program object. These parameters along with the unique vertex name form a closure that is sent to a remote process for execution.

3.3. Adding graph edges

New edges are created by applying a composition operation to two existing graphs as shown in Figure 3. In this figure, Circles are vertices and arrows are graph edges. A triangle at the bottom of a vertex indicates an input and one at the top indicates an output. Boxes (a) and (b) demonstrate cloning individual vertices using the carat operator. The two standard connection operations are pointwise composition using >= shown in (c) and complete bipartite composition using » shown in (d). (e) illustrates a merge using ||. The second line of the figure shows more complex patterns. The merge in (g) makes use of a "subroutine" from (f) and demonstrates a bypass operation. For example, each A vertex might output a summary of its input to C which aggregates them and forwards the global statistics to every B. Together the B vertices can then distribute the original dataset (received from A) into balanced partitions. An asymmetric fork/join is shown in (h) [6].

4. FEATURES

The primary features of Dryad are fault tolerance and dynamic modification of graph during run-time.

4.1. Fault Tolerance

When a vertex execution fails, the job manager will be notified. If the vertex reported an error cleanly/crashes then it is forwarded by the daemon to the job manager. If the daemon also fails then job manager receives a heartbeat timeout. The failed vertex A is re-executed. If vertex A's inputs fail, all upstream vertices are re-executed. vertex A is running slower than it's peers then creates duplicate executions and first output is used [5].

4.2. Dynamic Graph Refinement

The application passes the initial graph at the beginning and records all the callbacks. The graphs can be modified during run-

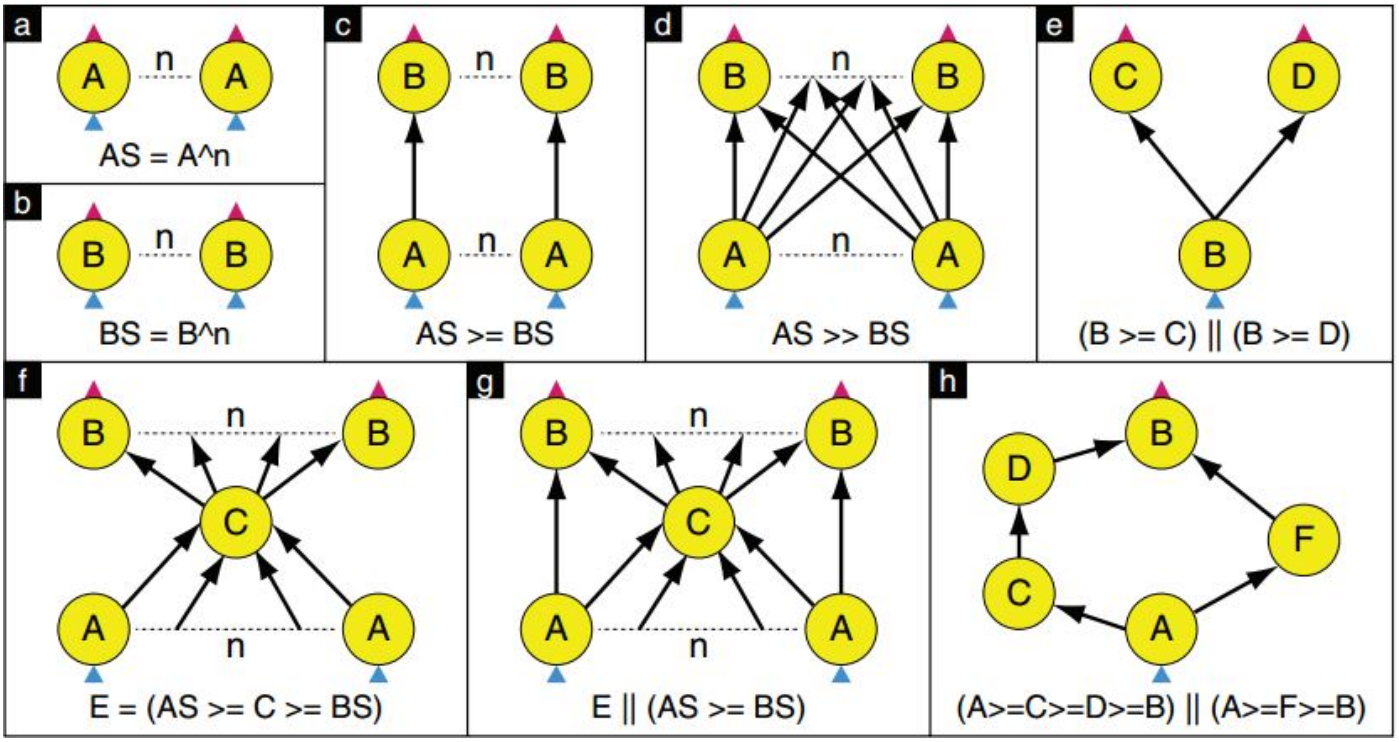


Fig. 3. The operators of the graph description language Source:[6]

time based on the output of the computations. However, there are some restrictions that do not allow to delete a vertex once it has been executed or alter the number and type of channels [5].

5. USE CASES

Dryad can be used for any large scale computational applications, Scientific applications, Large-data processing applications-indexing, search,etc. High-level language compilers use Dryad as a run-time. For example, SCOPE (Structured Computations Optimized for Parallel Execution) and DryadLINQ [7]. Professor of informatics at Indiana University, Geoffrey Fox had used Dryad and DryadLINQ to analyze RADAR data focused on glaciers to earn more about the earth's past and its present in order to make more informed, potentially life-saving predictions about its future [8].

6. PERFORMANCE

Computers	1	2	3	4	5	6	7	8	9
SQLServer	3780								
Two-pass	2370	1260	836	662	523	463	423	346	321
In-memory						217	203	183	168

Fig. 4. Time taken to process SQL query Source:[9]

The performance of Dryad was experimentally evaluated through two processes. In the first a relatively simple SQL query was distributed among 10 computers in the Dryad system and it performance was compared with that of a traditional commercial SQL server. The second is a map-reduce data-mining operation applied to 10.2 TBytes of data using a cluster of around 1800 computers [9]. The experiments were run in Microsoft Research laboratory with computers having 2 dual-core Opteron processors running at 2 GHz, 8 GB of DRAM and 4 disks(400 GB). Network connectivity was by 1 Gbit/sec Ethernet links connecting into a single non-blocking switch. Figure 3 summarizes the time in seconds to process an SQL query using different numbers of computers [9].

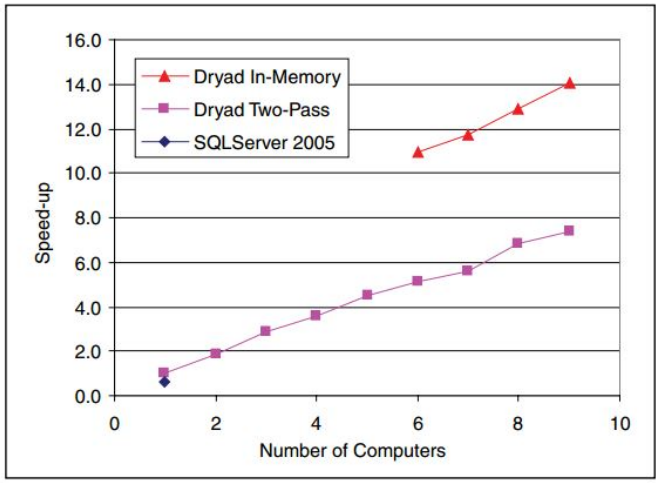


Fig. 5. speedup of the SQL query Source:[9]

From Figure 5 we can see that the speedup of the SQL query computation is nearly linear in the number of computers used. The baseline is relative to Dryad running on a single computer and times are as given in Figure 4 [9].

7. CONCLUSION

Microsoft Dryad are suitable only for applications that take the form of a directed acyclic graph. Dryad assumes that the vertices are deterministic and will fail if the application contains non-deterministic vertices. Since the job manager assumes that it has exclusive control over the computers in the cluster, it is difficult to efficiently run more than one job at a time Dryad [9]. It focuses only on throughput and does not improve the latency. In November 2012, Microsoft shifted their focus to Hadoop rather than improvising.

REFERENCES

[1] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. Lisboa-Portugal: ACM, Mar. 2007, p. 59, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf

[2] C. Poulain, "Dryad - microsoft research," Web Page, Mar. 2007, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/project/dryad/

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. Lisboa-Portugal: ACM, Mar. 2007, pp. 59–72, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf

[4] ——, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. Lisboa-Portugal: ACM, Mar. 2007, pp. 60–61, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf

[5] M. Isard, "Dryad - google tech talks," Web Page, Nov. 2007, accessed: 2017-2-24. [Online]. Available: https://www.youtube.com/watch?v=WPhE5JCP2Ak

[6] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. Lisboa-Portugal: ACM, Mar. 2007, pp. 63–64, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf

[7] Wikipedia, "Dryad (programming)-wikipedia," Web Page, Nov. 2016, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/blog/dryad-and-dryadlinq-academic-accelerators-for-parallel-data-analysis/

[8] D. Campbell, "Dryad and dryadlinq: Academic accelerators for parallel data analysis," Web Page, Feb. 2010, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/blog/dryad-and-dryadlinq-academic-accelerators-for-parallel-data-analysis/

[9] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. Lisboa-Portugal: ACM, Mar. 2007, pp. 71–72, accessed: 2017-2-24. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2007/03/eurosys07.pdf

Apache Mahout

NAVEENKUMAR RAMARAJU[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors:naveenkumar2703@gmail.com*

project-000, March 26, 2017

Apache Mahout[1] is an extensible programming environment to build scalable machine learning algorithms. It has algorithms to work in tandem with frameworks like Hadoop, Spark, Flink and H2O which specializes on dealing with large scale data. Samsara is a vector math environment to do linear algebra operations using distributed computing.

Keywords: Mahout, Samsara, Apache, Scalable, Machine learning

https://github.com/naveenkumar2703/sp17-i524/paper2/S17-IR-2029/report.pdf

1. INTRODUCTION

Mahout is an open source software to create scalable machine learning models. The initial version of Mahout targeted to implement all ten machine learning algorithms discussed in Andrew Ng's paper "Map-Reduce for Machine Learning on Multicore"[2] with scalability in mid. Further releases of Mahout added various implementations of clustering, classification, collaborative filtering and genetic algorithms in Java for usage in single machine as well as in clusters using map reduce.

As the popularity of in memory softwares like Spark started gaining popularity over disk based softwares like Hadoop, new features rolled out by Mahout did not support Map reduce. The module to do algebraic operations under the code "Samsara" is released in version 0.10 was compatible only with frameworks like Spark, H2O and Flink but not map reduce based frameworks. Samsara was mainly written in Scala and is optimized to operate well independent of the background. Another key feature of Samsara is that it supports R-like syntax for linear algebra operations.

Some of the algorithms available in map reduce stack were introduced for memory based frameworks as well and many of the implementations available in map reduce is yet to be introduced to other engines. A full list of algorithms available in Mahout and supported frameworks is discussed in section 2.

2. FEATURES

Mahout supports wide range of machine learning algorithms like classification, collaborative filtering and clustering, dimensionality reduction techniques like SVD, PCA and QR decomposition. Mahout Samsara environment provides linear algebra and statistics operations. Each of these are described in this section.

2.1. Samsara - Math Environment

Mahout Samsara[3] is a math environment to create and perform various math and linear algebra operations. Some of the key functionalities are BLAS (Basic Linear Algebra Subprogram), distributed row matrix, distributed ALS (Alternating Least Squares), PCA (principal component analysis), incore and distributed SPCA (Stochastic PCA), SVD (singular value decomposition), incore and distributed SSVD (Stochastic SVD), Eigen decomposition, Cholesky decomposition and similarity analysis.

One of the main advantage of Samsara is it supports R and Matlab like syntax using Scala's DSL (domain specific language) feature. DSL's are syntactic sugars for easy interpretation. An example is provided in section 5. Mahout Samsara is supported on Spark, H2O and Flink engines. Samsara is not available in Hadoop and map reduce based engines but has a different implementation that supports all these math operations.

2.2. Classification

Mahout has logistic regression using stochastic gradient descent, naive Bayes, complementary naive Bayes, random forest and hidden markov algorithms for classification. Naive Bayes available in Spark and map reduce is the only distributed classification algorithm. Others are supported only on single machines.

2.3. Clustering

Mahout supports clustering algorithms like K-means, fuzzy k-means, streaming k-means, spectral clustering and canopy clustering. These are available only for single machines and map reduce based environments.

2.4. Collaborative Filtering

Mahout has implementation for user based and item based collaborative filtering algorithms for single machine, map reduce

and spark engines. It also has implementation of matrix factorization with ALS, weighted matrix factorization using SVD for single machine and map reduce.

2.5. Other

Mahout supports several other features like Latent Dirichlet Allocation, row similarity job, collocations, sparse TF-IDF vectors from text, XML Parsing, Email Archive Parsing for map reduce engines. It also has Evolutionary Processes/Genetic algorithms implementation that runs on single machine.

2.6. GPU Acceleration

Mahout has capability to take advantage of GPU to accelerate matrix and vector operators on the backend using Vienna CL, OpenCL/CUDA.

3. LICENSING

Apache Mahout is an open source software available for free commercial usage and is licensed under Apache License, Version 2.0[4]. Associated frameworks like Spark, Hadoop, Flink and H2O are also open source products.

4. ECOSYSTEM

Mahout can be used with wide variety of distributed systems like Spark, H2O, Flink and Hadoop. It can also be used on single machine. A key benefit of Mahout is that the machine learning or math code can be used and written in same syntax independent of backends.

An illustration of how Mahout works by using Scala DSL for math operations with various engines is illustrated in figure 1.

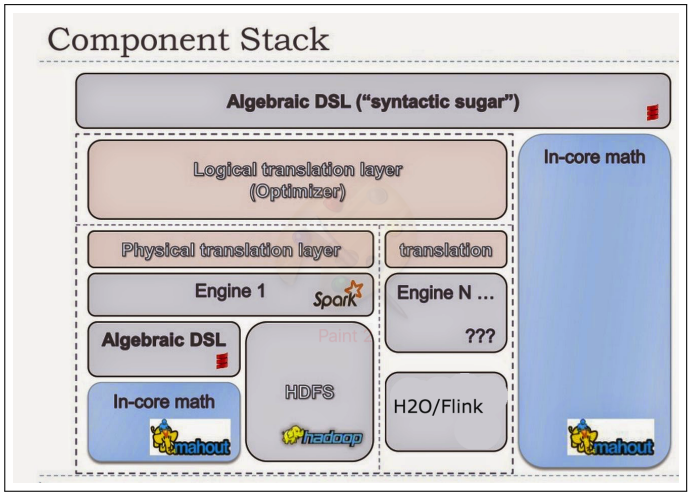


Fig. 1. Mahout Ecosystem.
Source: [5]

5. USE CASES

An use case for recommendation system and a simple linear algebra operation is provided in this section.

5.1. Recommendation system with Spark Engine

A recommendation system is used to recommend most relevant product that he/she might be interested and boost the sales in online platforms. Recommendation can be done based on user similarity or item similarity. In case of user similarity, recommendation is provided based on idea that users with similar interest would buy similar products. Similar users are identified, grouped and recommendations are made based on difference in set of products. Where as item similarity is based on identifying the products that were bought together and recommending the products that is not in customer's basket or purchase history. Recommendation based on item similarity is very popular in industry as number of product types is way less than number of customers with regular purchase patterns in large retail industry.

An illustration of how Mahout can be used with Spark for item based recommendation is illustrated in figure 2.

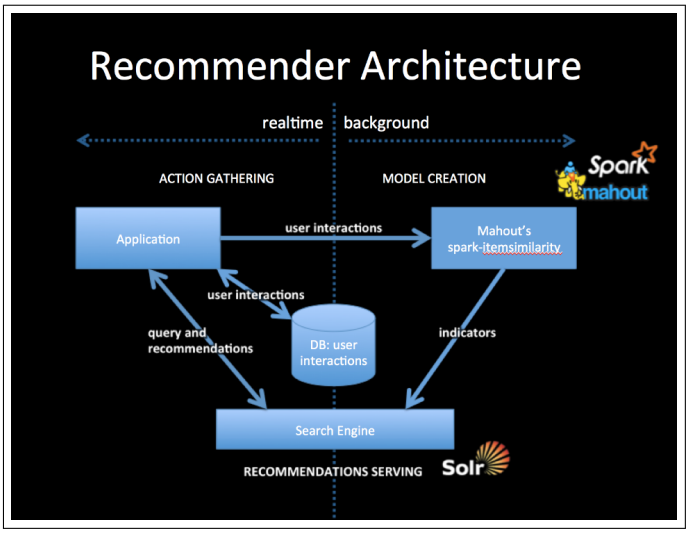


Fig. 2. Recommender Illustration.
Source: [6]

Mahout's mapreduce version of item similarity takes a text file that is expected to have user and item IDs and provide recommendations based on content.

5.2. SPCA

The equation to compute stochastic principal component analysis is given in equation 1

$$G = BB' - C - C' + s_q s_q' \xi' \xi \quad (1)$$

Mahout code to compute this using Scala DSL is illustrated in algorithm 1,

Algorithm 1. Mahout DSL to compute SPCA

```
1  val g = bt.t %*% bt - c - c.t + (s_q cross s_q)
2                  * (xi dot xi)\
```

This could be used in single or distributed machine in Spark, H2O or Flink to perform SPCA.

6. USEFUL RESOURCES

Apache Mahout Cookbook[7] by Piero Giacomelli is a good introductory book on Mahout. Apache Mahout Beyond MapReduce[8] by Dmitriy Lyubimov provides an exhaustive coverage of Mahout Samsara and math operations.

Mahout website[9] has a compilation of useful resources like books, tutorials and talks about Mahout and machine learning.

7. CONCLUSION

Apache Mahout provides an open source environment with scalable algorithms for machine learning and math operations using DSL. It can be used with multiple backends like Spark, Scala, Flink and Hadoop. Same code can be used for single machine and distributed computing of algorithms in any supported backends.

ACKNOWLEDGEMENTS

This work was done as part of the course "I524: Big Data and Open Source Software Projects" at Indiana University during Spring 2017. Thanks to our Professor Gregor von Laszewski and associate instructors for their help and support during the course.

REFERENCES

[1] Apache Software Foundation, “Apache mahout,” Web Page, Jun. 2016, accessed: 2017-3-26. [Online]. Available: http://mahout.apache.org/

[2] C. tao Chu, S. K. Kim, Y. an Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng, “Map-reduce for machine learning on multicore,” in *Advances in Neural Information Processing Systems 19*, P. B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 281–288, accessed: 2017-3-26. [Online]. Available: http://papers.nips.cc/paper/3150-map-reduce-for-machine-learning-on-multicore.pdf

[3] Apache Software Foundation, “Mahout samsara incore references,” Web Page, May 2016, accessed: 2017-3-26. [Online]. Available: http://mahout.apache.org/users/environment/in-core-reference.html

[4] Apache Software foundation, “Apache License 2.0,” Web Page, Jan. 2004, accessed: 2017-3-26. [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0

[5] D. Lyubimov and A. Palumbo, “Mahout 0.10.x: first mahout release as a programming environment,” Web Page, Apr. 2015, accessed: 2017-3-26. [Online]. Available: http://www.weatheringthroughtechdays.com/2015/04/mahout-010x-first-mahout-release-as.html

[6] Apache Software Foundation, “Intro to cooccurrence recommenders with spark,” Web Page, May 2016, accessed: 2017-3-26. [Online]. Available: http://mahout.apache.org/users/algorithms/intro-cooccurrence-spark.html

[7] P. Giacomelli, *Apache Mahout Cookbook*. Packt Publishing, 2016, accessed: 2017-3-26.

[8] D. Lyubimov and A. Palumbo, *Apache Mahout: Beyond MapReduce*. Createspace Independent Publishing Platform, 2016, accessed: 2017-3-26.

[9] Apache Software Foundation, “Mahout book, tutorials, talks,” Web Page, May 2016, accessed: 2017-3-26. [Online]. Available: https://mahout.apache.org/general/books-tutorials-and-talks.html

Neo4J

SOWMYA RAVI[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: sowravi@iu.edu.com*

project-000, March 26, 2017

Neo4J is a graph database designed for fast data access and management. The data is stored in the form of nodes and relationships in Neo4J. The unique approach it takes to store data makes is far more efficient compared to relational databases when the number of relationships within the data increases. Moreover, it has the ability to store trillions of data entries in a compact manner. Neo4J comes along with Cypher, a highly readable querying language. The paper elaborates the clustering activities used by Neo4j to achieve distributed computing its uses [1].

Keywords: Cloud, I524

https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf

1. INTRODUCTION

Certain problems present in the world cannot be solved by using relational databases. For e.g. a Social graph representing a the network of friends in a social networking website. In this case the number of relationships in the data is too extensive and the relational databases perform poorly. Graph data bases on the other hand make the task of storing huge amounts of data relatively simple and efficient. Neo4J is one such NoSQL, graph database which was developed to be used in the kind of problems mentioned before [2]. Fig.1 illustrates a simple social network graph.

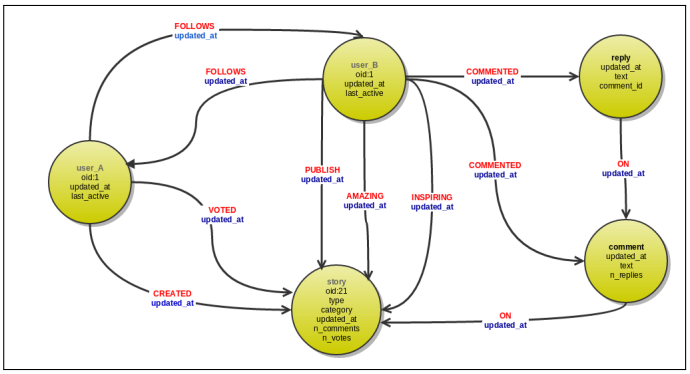


Fig. 1. A Social Network graph [3]

Neo4j is an open source data management software. At its core, Neo4j stores data in the form of nodes and relationships. It is often deployed in a production environment as a fault tolerant cluster of machines. The high scalability and slow traversal times make it far more efficient than the conventional relational databases [1].

2. CYPHER PROGRAMING LANGUAGE

Neo4j uses its own programming language, Cypher, for data creation as well as querying. Cypher is capable of doing SQL like actions. In addition, it can specifically perform a powerful query called traversals. Traversal involves moving along a specific set of nodes in the database thereby tracing a path. This allows to leverage the spatial structuring of the data to get valuable information, similar to network analysis [4].

3. CLUSTERING FOR THE ENTERPRISE

This section discusses Neo4js architecture with respect to clustering. Neo4j uses clustering of machines to achieve high throughput, availability and disaster recovery [5]. Neo4j offers two kind of clustering

1. Causal Clustering
2. Highly Available clustering

3.1. Causal Clustering

The Causal clustering of machines in Neo4j is aimed at providing two important features [6]

1. **Safety**: The core servers of Neo4j ensure fault tolerance.
2. **Scalability**: Achieved using Read replicas that make massive scaling possible.

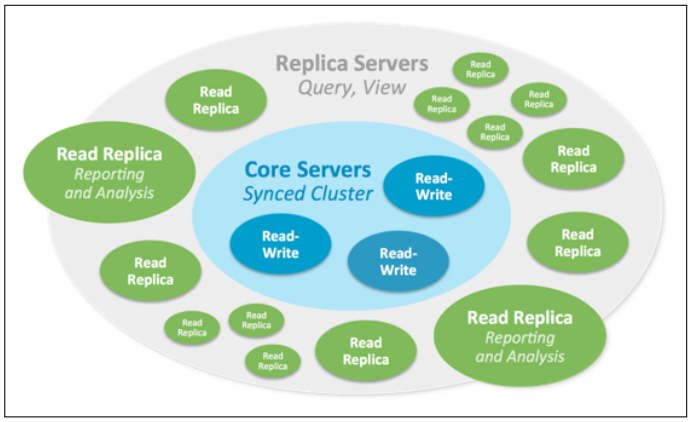


Fig. 2. Architecture of Causal Clustering [6]

The architecture of causal clustering is shown in Fig.2.

For operational purposes, the cluster is usually separated into two components: the core servers and the read replicas.

3.1.1. Core Servers

The Core servers are responsible for safe data storage. This is achieved by replicating all incoming queries/transactions using Raft protocol (A log replication protocol) [6]. The protocol ensures the durability of data before committing to the query request. Usually, a transaction is accepted only when a majority of the servers,calculated as $N + 1/2$, have accepted it. This number is directly proportional to the number of core servers N. Hence, as the number of core servers grows, the size of majority required for committing to an end user also increases. increases [6].

In practice few machines in the core server cluster is enough to provide fault tolerance. This number is calculated using the formula: $N = 2F + 1$ where N is the number of servers required to tolerate F faults [6]. When a core server suffers a large number of faults, it is automatically converted to a read-only server for safety purposes.

3.1.2. Read replicas

Read Replicas are Neo4j databases that scale out the incoming queries and procedures. They act like cache memories to the core servers which safeguard the data. Even though the read replicas are full-fledged databases, they are equipped to perform arbitrary read-only activities [6].

Read Replicas are created asynchronously by core servers through log-shipping [6]. Log shipping occurs when the read replicas poll the core servers for new transactions and the transactions are shipped from the core severs to the read replicas. This polling occurs periodically. Usually, a small number of core servers ship out queries to a relatively large number of read replicas, allowing a large fan out of workload thereby, achieving scalability [6]. The read replicas unlike the core servers do not participate in deciding the cluster topology.

3.1.3. Causal Consistency

In applications, data is generally read from a graph and written to a graph. In order to ensure the causal consistency in the data, the write operation must take into account previous write operations. The Causal Consistency model for distributed computing requires every node in the system to see causally related operations in the same order. This model ensures that the data can be written to cores and the written data be read from read replicas. Fig.3 illustrates a Causal Cluster with causal consistency

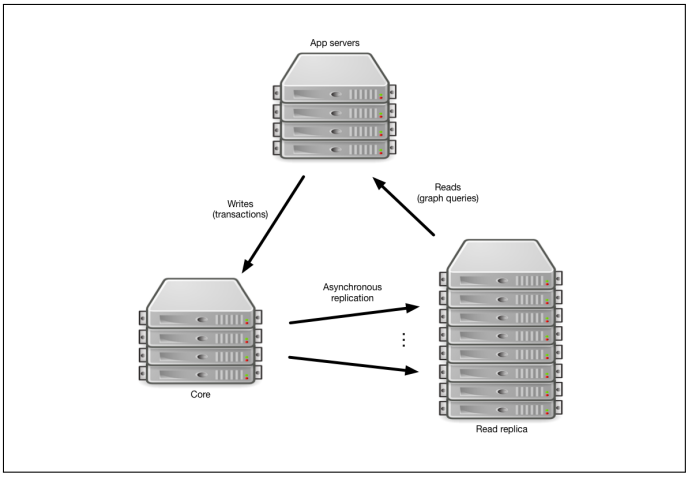


Fig. 3. Causal Cluster with causal consistency set via Neo4j drivers [6]

4. HIGHLY AVAILABLE CLUSTER

In this type of cluster each instance of the cluster contains full copy of the data in their local database. The cluster can be visualized as containing a single master with multiple slaves in which each instance is connected to every other instance(A 3 member cluster is shown in Fig.4.0)

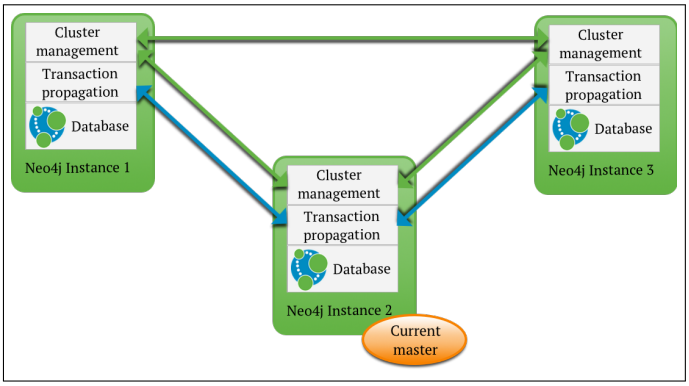


Fig. 4. A Highly Available cluster model [7]

Also, each instance contains the logic to perform read/write operations and election management [7]. Every slave, excluding the Arbiter instance periodically communicate with the master to keep databases up to date [7]. There is a special slave called the Arbiter explained in the following section.

4.0.1. Arbiter Instance

The Arbiter instance is a special slave that participates in cluster management activities but does not contain any replicated data. It simply contains a Neo4j software running in arbiter mode [7].

4.0.2. Transaction Propagation

Write Transactions performed directly on the Master will be pushed to slaves once the transaction is successful. When a write transaction is performed on a slave, the slave synchronizes

with the Master after each write operation. The write operation on slave is always performed after ensuring that the slave is synchronized with the Master [7].

4.0.3. Failover

When an instance becomes unavailable, it is marked as temporarily failed by other instances. If the Master fails then, another member in the cluster will be elected as the Master [7].

5. USE CASES

Some of the use case of Neo4j is given below [8].

- Fraud Detection
- Graph based search
- Network and IT operations
- Real-Time Recommendation system
- Social Network
- Identity and Access Management

5.0.1. Neo4j for Social Network Analysis

Social Networks are already graphs and several possible use cases for Social Networks are listed below [9].

- The Friends of Friends recommendation in social networks is one useful use case. The traversal capability makes this task simple and efficient.
- It can be used to discover previously unknown relationships in massive networks. People get connected through multiple channels. Neo4j may be of great help in studying these relationships
- Collaboration and Sharing become far more easier in the presence of graph databases. The clustering facility enables data being safe and secure yet highly available. Content visibility increases to a great extent.

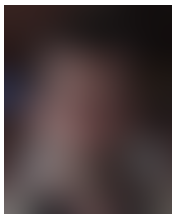
6. CONCLUSION

Neo4j being an open source, graph based and a highly scalable software, it is suitable for applications that deal with huge amounts of data. Also, Neo4j can be integrated with other tools and software such as Spark, Docker, Elastic Search, MongoDB etc. The versatility of Neo4j makes it a great software aid for data scientists trying to analyze relationships and networks in real time as well as in batch.

REFERENCES

[1] Neo4j, "Chapter 1. introduction," Web Page, last Accessed: 03.24.2017. [Online]. Available: https://neo4j.com/docs/operations-manual/current/introduction/
[2] S. Haines, "Introduction to neo4j," Web Page, last Accessed: 03.24.2017. [Online]. Available: http://www.informit.com/articles/article.aspx?p=2415370
[3] M. project, "Social network project," Web Page, last Accessed: 03.24.2017. [Online]. Available: https://meu-solutions.com/case-studies-social-network-project/
[4] R. Ostman, "Graphical database of citation network analysis," PDF Document, last Accessed: 03.24.2017. [Online]. Available: http://webdocs.ischool.illinois.edu/crt/ostman.pdf
[5] Neo4j, "Chapter 4. clustering," Web page, last Accessed: 2017.02.24. [Online]. Available: https://neo4j.com/docs/operations-manual/current/clustering/
[6] ——, "Causal cluster," Web page, last Accessed: 2017.03.24. [Online]. Available: https://neo4j.com/docs/operations-manual/current/clustering/causal-clustering/introduction/
[7] ——, "Highly available cluster," Web page, last Accessed: 2017.03.24. [Online]. Available: https://neo4j.com/docs/operations-manual/current/clustering/high-availability/architecture/
[8] ——, "Graph database use cases," Web page, last Accessed: 2017.03.24. [Online]. Available: https://neo4j.com/use-cases/
[9] ——, "Solutions: Social network," Web page, last Accessed: 2017.03.24. [Online]. Available: https://neo4j.com/use-cases/social-network/

AUTHOR BIOGRAPHIES



Sowmya Ravi pursuing Masters in Data Science from Indiana University. Her research interests include Machine Learning, Data Mining and Big Data Analytics

OpenStack Nova: Compute Service of OpenStack Cloud

KUMAR SATYAM[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: ksatyam@indiana.edu*

paper-2, March 27, 2017

OpenStack Nova is the compute service of the OpenStack cloud system.It is designed to manage and automate the pools of computer resources and can work on bare metal and high performance computing. It is written in python.We will discuss the main components included in the Nova Architecture [1].

Keywords: Cloud,OpenStack,Nova, API, I524

https://github.com/satyamsah/sp17-i524/blob/master/paper2/S17-IR-2031/report.pdf

INTRODUCTION

Nova is responsible for spawning vm instances in openstack environment. It is built on a messaging architecture which runs on several servers. This architecture allows components to communicate through a messaging queue. Nova together shares a centralized SQL-based database for smaller deployment. For large deployments an aggregation is used to manage data across multiple data stores[2].

NOVA FRAMEWORK

Nova is comprised of multiple server processes, each performing different functions. The OpenStack provided user interface for Nova which is a REST API. During invocation of the API, the Nova communicates via RPC (Remote procedure call) passing mechanism.

The API servers process REST requests, which typically involve database reads/writes. RPC messaging is done via the 'oslo.message' library. Most of the nova components can run on different servers and have a manager that is listening for RPC messages. One of the components is Nova Compute where a single process runs on the hypervisor it is managing.

Nova has a centralized database that is logically shared between all components[4].

NOVA COMPONENTS

Below are the major components of Nova:

DB: An SQL database for data storage.This is the SQLAlchemy-compatible database. The database name is 'nova'. The 'nova-conductor' service is the only service that writes to the database.

ARCHITECTURE -OPENSTACK NOVA

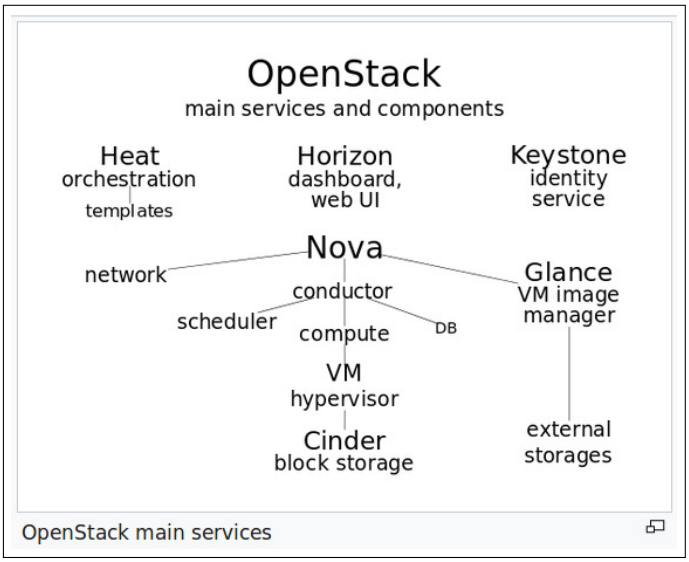


Fig. 1. Architecture of OpenStack cloud using Nova [3]

API: It is the component that receive HTTP requests, converts commands and communicates with other components via the 'oslo.messeging' queue or HTTP. The 'python-novaclient 7.1.0' is a python binding to OpenStack Nova API which acts as a client for the same. The nova-api provides an endpoint for all API queries (either OpenStack API or EC2 API), initiates most of the orchestration activities (such as running an instance) and also enforces some policy (mostly quota checks)

Schedular: The 'nova-schedular' is a service to determine how to dispatch compute requests.For example, it determines on which host a VM should launch. Compute is configured with a default scheduler options in the /etc/nova/nova.conf file[5].

Network: It manages IP forwarding, bridges, and vlans. The network controller with nova-network provides virtual networks to enable compute servers to interact with each other and with public network. Compute with nova-network support the following modes, which are implemented as Network Manager types:

- Flat Network Manager
- Flat DHCP Network Manager
- VLAN Network Manger

volume : It manages creation, attaching and detaching of persistent volumes to compute instances. This functionality is being migrated to Cinder, a separate OpenStack service

Compute: It manages communication with hypervisor and virtual machine

Conductor: It handles requests that need coordination, acts as a database proxy, or handles object conversions

MAJOR TASK PERFORMED BY OPENSTACK NOVA SERVICE

Below are the major tasks performed by Nova Service[6]

- Authenticating against the nova-api endpoint
- Listing instances
- Retrieving an instance by name and shutting it down
- Booting an instance and checking status
- Attaching Floating IP address
- Changing a security group
- Retrieving console login

NOVA AND OTHER OPENSTACK SERVICE

Nova is the main Openstack services as it interact with all the services to set IAAS stack. Nova interact with Glance service to provided images for VM provisioning.It also interact with the Queue service via a nova-scheduler and nova-conductor API.It interacts with Keystone for authentication and authorization services. It also interacts with Horizon for web interface.

NOVA DEPENDENCE ON AMQP

AMQP is the messaging technology chosen by the OpenStack cloud.The AMQP sits between any two Nova components and allow them to communicate in a loosely coupled fashion. Nova Components use RPC to communicate to one another.It is build on the pub/sub paradigm to have the benefits. Decoupling between client and server(such that the client does not need to know where the servant's reference is) is a major advantage of AMQP[7].

ORCHESTRATION TASK IN NOVA

Nova-Conductor service plays an important role to manage the execution of workflows which involve the scheduler. Rebuild, resize, and building the instance are managed here. This was done in order to have better separation of responsibilities between what compute nodes should handle and what schedular should handle and to clean up the path of execution. In order to query the schedular in a synchronous manner it needed to happen after the API had returned a response otherwise API response times would increase and thats why conductor was chosen. And changing the schedular call from asynchronous to synchronous helped to clean up the code[8].

The earlier logic was complicated and the scheduling logic was distributed across the code.The earlier process was changed to the following:

- API receives request to build an instance.
- API sends an RPC cast to conductor to build an instance.
- Conductor sends an RPC call to schedular to pick a compute and waits for the response. If there is a schedular fail, it stops the build at the conductor.
- Conductor sends an RPC cast to the compute to build the instance. If the build succeeds, stop here. If it fails then the compute sends an RPC cast to conductor to build an instance. This is the same RPC message that was sent by the API.

OPENSTACK NOVA SUPPORT

Earlier Openstack was supported on KVM but its support has been extended QEMU. Microsoft Hyper -V and Vmware ESXi too provide extended support. Nova has support for XenServer and XCP through XenAPI virtual layer.It also support bare metal deployment and provisioning from 'Ironic' version. This means it is possible to deploy virtual machines. By default, it will use PXE and IPMI to provision and turn on/off the machine. But from the Ironic version it support vendor-specific plugins which may implement additional functionality.

OPENSTACK NOVA IN BIGDATA

A use case has been developed to leverage OpenStack to perform big data analytics. OpenStack used cassandra database for columnar data structure. PostgreSQL for relational data structure.This use case also allows us to configure Hadoop distributed file system for large unstructured data. OpenStack Sahara has come up with core cloud components of Big data[9].

OPENSTACK NOVA AND OTHER COMPETITORS

The OpenStack nova does the same task as it is being done by AWS EC2, Google CE and Microsoft Azure VM. With respect to Beach marking the main difference is the cloud administrator can upload their images in OpenStack where as in AWS and Google Cloud storage, one need to use the pre-defined list.The AWS is used mainly as public cloud where as the Openstack can be used a private cloud. But OpenStack has an advantage of customizing our own cloud configuration which is not there in any of the vendor specific clouds.

CONCLUSION

We discussed about the main components of OpenStack Nova and how it is interacting with different other Openstack services.We also showed use case of running big data problem on Openstack. We also discussed the compute service offerings provided by cloud vendors other than OpenStack.

REFERENCES

[1] Wikipedia, "Openstack nova wikipedia," accessed: 03-23-2017. [Online]. Available: https://en.wikipedia.org/wiki/OpenStack

[2] OpenStack, "Openstack nova official," accessed: 03-23-2017. [Online]. Available: https://docs.openstack.org/developer/nova/architecture.html

[3] Wikipedia, "Openstack nova bigdata," accessed: 03-23-2017. [Online]. Available: https://en.wikipedia.org/wiki/OpenStack

[4] P. Website, "Openstack nova on pepple website," accessed: 03-23-2017. [Online]. Available: http://ken.pepple.info/openstack/2011/04/22/openstack-nova-architecture/

[5] O. Website, "Openstack nova schedular," accessed: 03-23-2017. [Online]. Available: https://docs.openstack.org/kilo/config-reference/content/section_compute-scheduler.html

[6] I. Developers, "Openstack nova ibm," accessed: 03-23-2017. [Online]. Available: https://www.ibm.com/developerworks/cloud/library/cl-openstack-pythonapis/

[7] O. Website, "Openstack nova amqp," accessed: 03-23-2017. [Online]. Available: https://docs.openstack.org/developer/nova/rpc.html

[8] ——, "Openstack nova orchestrator," accessed: 03-23-2017. [Online]. Available: https://docs.openstack.org/developer/nova/conductor.html

[9] ——, "Openstack nova bigdata," accessed: 03-23-2017. [Online]. Available: https://www.openstack.org/summit/san-diego-2012/openstack-summit-sessions/presentation/big-data-on-openstack-a-rackspace-use-case

Heroku

YATIN SHARMA[1,*,+]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: yatins@indiana.edu
[+] HID - S17-IR-2034

Paper-002, March 26, 2017

Heroku is a cloud application program that enables developers to build and deploy application in almost any language. It provides everything a developer would need to build customer facing application.The core of Heroku is called Dyno, which can be thought of as virtual machine that runs just your application.Dynos are scalable according to our needs as well.

Keywords: Heroku, PaaS, Dyno, Dyno Manifold, Logplex, Toolbeit client, Procfile.

https://github.com/yatinsharma7/sp17-i524/tree/master/paper2/S17-IR-2034/report.pdf

INTRODUCTION

Heroku[1] is Platform as a Service(Paas)[2] that enables us to build application in any language on demand. It allows us to deploy web applications seamlessly as well as monitor and share with other developers at the click of a button. It is all about rapid application development for the cloud, using the underlying platform infrastructure and software add-ons to build, deploy and monitor large and scalable web application. At present it handles 5 billion requests per day.

BENEFITS

Heroku is language agnostics and provides great flexibility in choosing an appropriate programming language to develop the web application. It has core support for Ruby, Java, Python, Node.js, and PHP. Also, any other language can be supported by using a feature called buildpacks. Heroku provides a lot of flexibility in managing our applications after deployment using the Heroku command line tool running on the client machine or on the Heroku Infrastructure. Heroku is Git[3] focused and makes it easy to share code using version controlling. Heroku is also a part of the Salesforce family. It has a feature called Heroku Connect that allows to build application that share data with the Salesforce deployment.

ARCHITECTURE

The Heroku architecture consists of platform stack containing various runtime libraries, OS, and underlying infrastructure.The unit of work in Heroku framework is called Dyno. It can be thought of as packaged running version of our code that the application interacts with. Dynos are responsible for receiving web requests, writing an output and connecting to application resources such as databases. They are fully isolated containers

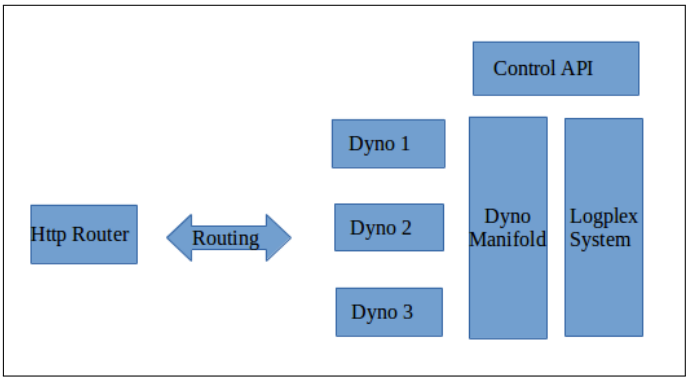


Fig. 1. High Level Architecture of Heroku platform

running on the Dyno Manifold, which is the building block for execution environment. Dyno Manifold is responsible for process management, isolation,scaling, routing, distribution that are necessary for to run the web and worker processes. It is also fault tolerant and distributed in nature. If any dynos fails, the manifold restarts them automatically, hence removing a lot of maintenance hassles.

Dyno Execution

Dynos are capable of serving many request per second and execute in complete isolation from each other. Each dyno gets its own virtual environment that it can use to handle its own client requests. Dynos use LXC to provide container like behavior to achieve complete isolation from one another.There is a memory restriction of 1.5 GB per dyno, beyond which the dyno is rebooted with a Heroku error which could lead to a memory leak.

Execution Flow

Process type is the declaration of the command that defines the structure to be used while instantiating a process. Heroku has two process types- web process, which is responsible for handling HTTP client requests and the worker process, which is responsible for executing other tasks such as customer jobs of running background jobs and queuing tasks.

Logging Architecture

Heroku logplex system provides a flexibility facility by giving us an overall view of our application runtime behavior. It forms the basis of the Heroku logging infrastructure. It routes log streams form various sources into single output(for example archival system). The logplex system keeps the most recent data (typically 1500 logs) that are important to extract relevant information from the application being run.

Http Routing

Routing Mesh are responsible for routing the web requests to the appropriate web process dyno. It is also responsible for seeking the application's web dynos within the dyno manifold and forwarding the HTTP web requests to one of them. The routing mesh uses a round robin algorithm to distribute the request acroos various dynos. Since the dynos could be running in distributed manner, the routing mesh manages the access and routing internally and none of the dynos are statically addressable. Heroku also supports multithreaded and asynchronous application, accepting many network connections to process client requests.

PROCESS ARCHITECTURE

A Heroku application can be thought of a multiple process, each consuming resources like a normal UNIX process, that run on the Heroku Dyno manifold. Heroku defines each process through configuration file called Procfile- which is a text file, placed in the root of the application and contains the format describing how our application will run.

HEROKU PLATFORM API

Heroku platform API is a tool that enables us to call Heroku platform services and create application, plug in new add-ons simply using HTTP. It gives the developers complete control over their application. The three components that define the behavior of the API are : 1)Security 2)Schema 3)Data. The client accesses the API using standard methods defined for HTTP. The API then acts on the request and returns the result in JSON format.

SECURITY

Heroku employs various measures to ensure that the application and data stores within the platform are secure from external attacks,thefts and hacks. Heroku enforces SSH[4] protocol to encrypt the source code while they are getting pushed into the Heroku environment. Any application that runs on Heroku is in complete isolation from each other, so that no two application can see each other getting executed. It also restricts applications from making local network connection between hosts. It enables data security by keeping the data in access controlled databases.

GETTING STARTED

There are few prerequisites that we need to perform before we can start using Heroku: 1) Get Heroku account 2) Install Heroku toolbeit client[5] 3) Set up SSH for our user account. Heroku toolbeit is the client software requird to work with the Heroku platform and contains the following component: Heroku Client, Foreman and Git.

CONCLUSION

Heroku is dramatically different from the traditional hosting or any normal cloud infrastructure offerings. Instead of thinking about virtual servers as individual units and trying to figure out how many do we need and the communication between them, Heroku completely abstracts the servers and filesystems away. As a developer we just have to push the code and heroku will manage all the rest of the process. Heroku provides a complete developer experience and application runtime and also frees the developer from hassles of underlying infrastructure. It manages all that in scalable and highly maintainable fashion.

ACKNOWLEDGEMENTS

The author thanks Prof. Gregor von Laszewski for his technical guidance.

REFERENCES

[1] "Cloud application platform | heroku," Web Page, online; accessed 13-Mar-2017. [Online]. Available: https://www.heroku.com/

[2] "Platform as a service - wikipedia," Web Page, online; accessed 13-Mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Platform_as_a_service

[3] "Github," Web Page, online; accessed 13-Mar-2017. [Online]. Available: https://github.com/

[4] "Secure shell- wikipedia," Web Page, online; accessed 17-Mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Secure_Shell

[5] "Heroku cli | heroku dev center," Web Page, online; accessed 17-Mar-2017. [Online]. Available: https://devcenter.heroku.com/articles/heroku-cli

D3.js

PIYUSH SHINDE[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: piyushsshinde1992@gmail.com*

Paper-2, March 27, 2017

Data Driven Documents (D3) is a open source JavaScript library used to create dynamic, interactive visualizations on a webpage. D3 uses HTML, SVG and CSS to create visualizations with a data-driven approach to Document Object Model (DOM) manipulation, enabling users to utilize the full capabilities of modern browsers and the freedom to design the right visual interface for their data [1]. This paper provides a brief introduction to D3 and its various features. It also discusses D3's use cases, advantages and limitations.

Keywords: Document Object Model, Data Visualization, Chart

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2035/report.pdf

1. INTRODUCTION

Several graphical forms can be used to envision large quantitative data sets such as graphs, charts, maps and diagrams. These data sets are increasing exponentially since the advent of the World Wide Web, making the task of efficient data visualization more challenging. As a result, web browsers are considered as ideal platforms for visualizing large data sets.

Designers often employ multiple tools simultaneously for building visualisations like HTML for page content, CSS for aesthetics, JavaScript for interaction, SVG for vector graphics [2]. One of the most important advantages is the uninterrupted cooperation between most of the technologies using the web as a platform, which is enabled by a document object model (DOM). The DOM reveals the ordered structure of a web page, aiding it's manipulations.

Data Driven Documents (D3) is an open source JavaScript library, which helps in efficient manipulation of the document object models based on data. It was released in 2011 by Heer, Ogievetsky and Bostock, then part of Computer Science Department of Stanford University, Stanford, CA 94305. D3 provides a toolkit for visualizing data using web standards such as HTML, SVG, and CSS.[1].

D3 resembles to document transformers such as jQuery that simplify the act of document transformation in web browsers. D3 uses the DOM's standard SVG syntax, that shows similarities with the graphical abstractions used in graphics libraries such as Processing and Raphaël. D3 is a generalization of Protovis, and through helper modules more complex visualizations can be achieved efficiently.

D3's main features include selections, transitions and update, enter and exit functions. We will glance through these features in the next section.

2. FEATURES

D3's basic operand is the selection. Operators act on selections, modifying content. Data joins bind input data to elements, producing enter and exit sub-selections for the creation and destruction of elements with respect to data. Animated transitions interpolate attributes and styles smoothly over time [2].

2.1. Selections

Selections allow the user to select and manipulate HTML elements in a very simple way. D3 adopts the W3C Selectors API that contain predicates to select elements by tag, class, unique identifier, attribute, containment, or adjacency [3]. Unique selection methods like union and intersection can be used on these predicates. Multiple operations can be performed after selecting an element by chaining the operations.

D3 provides select and selectAll methods for single and multiple element selections. The former selects only the first element that matches the predicates, while the latter selects all matching elements in document traversal order [2].

2.2. Data

Styles, attributes, and other properties are represented as functions of data in D3. They are simple, but powerful. D3 provides many built-in reusable functions and function factories, such as graphical primitives for area, line, and pie charts.

The data operator binds input data to selected nodes. Data is specified as an array of values such as numbers, strings or objects, and each value is passed as the first argument, along with the numeric index to selection functions. By default, data is joined to elements by index, the first element in the data array is passed to the first node in the selection, the second element to

the second node, and so on. Once the data has been bound to the document, you can omit the data operator, D3 retrieves the previously-bound data. This allows you to recompute properties without rebinding [3].

2.3. Enter, Update and Exit

D3's Enter and Exit selections, aid users to create new nodes for incoming data and remove outgoing nodes that are no longer required.

When data is bound to a selection, each element in the data array is paired with the corresponding node in the selection. In case of fewer nodes as compared to elements of the data array, the extra data elements form the enter selection, that can be instantiated by appending to the enter selection.

Updating nodes are the default selection—the result of the data operator [3]. In case of skipped enter and exit selections, D3 automatically selects only the elements for which corresponding data exists. Properties that are constant for the life of the element are set once on enter, while dynamic properties are recomputed per update [2]. The initial selection can be divided into three parts: the updating nodes to modify, the entering nodes to add, and the exiting nodes to remove. Handling these three cases separately, precisely define the operations that run on each node, thereby optimizing performance and offering greater control over transitions.

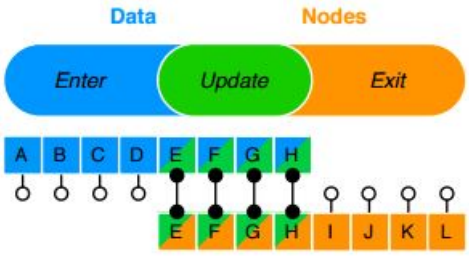


Fig. 1. New data (blue) joining old nodes (orange) results in three sub-selections: enter, update and exit [3].

2.4. Transitions

D3's focus on transformation extends naturally to animated transitions. Transitions gradually interpolate styles and attributes over time. D3's interpolators support both primitives, such as numbers and numbers embedded within strings (font sizes, path data, etc.), and compound values. D3's interpolator registry can even be registered to support complex data structures.

D3 reduces overhead by adjusting only the attributes that change and allows greater graphical complexity at high frame rates. D3 allows sequencing of complex transitions via events. D3 does not replace the browser's toolbox, but exposes it in a way that is easier to use [3].

3. USE CASES

The examples tab of the official D3 website displays more than 400 examples of data visualizations, built using D3 [4]. Examples include visualizations for newspapers, games, libraries and tools suggesting D3's reliability and ability to transform even the most complex data into clear visualizations. These visualizations run interactively inside web browsers and are available for anyone who wants to visualize data.

Many of them are accompanied by a full documentation of the steps to manipulate documents based on data. The most commonly used types of data visualization are histogram, area chart, line chart, multi series line chart, bar chart, scatter-plot, pie chart, graphs, trees and maps [4]. D3 also shows visualizations updating in real time.

Here are three real world examples of simplified data visualizations using D3.

3.1. Wimbledon 2013 Data Visualisations

This example displays a series of ten different data visualizations of the Wimbledon tennis tournament created by Peter Cook [5]. The original data was collected from British tennis data website [6].

Amongst the 10 visualizations, 3 of them include a circular match tree, a bubble chart and a horizontal histogram. The circular match tree displays concentric circles labelled as the different rounds of the Wimbledon 2013. Each concentric circle displays the player names. The result was of each match in each round is displayed by hovering the mouse over the names. The winner's name is displayed in the center of the circle.

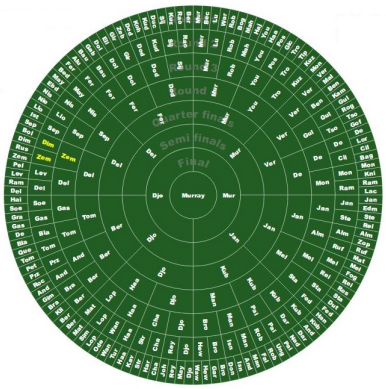


Fig. 2. Wimbledon 2013: Circular Match Tree [7].

The bubble chart displays pair of bubbles connected by an arrow. Each arrow indicates a match where the winner had a lower ATP ranking than the loser. The size of each bubble is proportional to the ATP points of the player. Hovering the mouse over a bubble displays both the players and their ATP points with the result of the match.

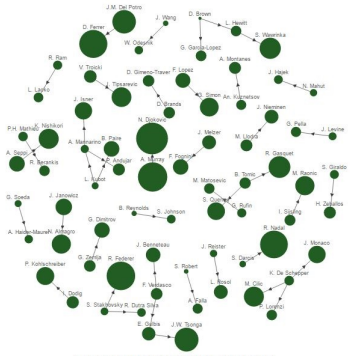


Fig. 3. Wimbledon 2013: Bubble Chart [8].

The histogram displays the list of the top 32 players of Wimbledon 2013. It displays the number of matches won, sets won,

games won and ATP ranking of all the 32 players by clicked on the respective tabs [9].

3.2. Earthquakes in Chile since 1900

This example helps us visualize the most important seismic events in Chile since 1900 [10]. The earthquake data was retrieved from the ANSS Composite Catalog and joined with the Centennial Catalog from the USGS.

A seismic event is displayed as a circle with it's occurrence year. The radius and the color of the circles are a function of the earthquake magnitude.

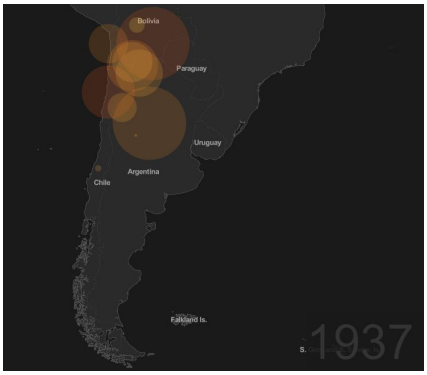


Fig. 4. Earthquakes in Chile since 1900: Map [10].

3.3. Visualizing the Racial Divide

This example displays a visualization that highlights the impact of the segregation that exists in many US cities today based on race using d3 and force-directed maps [11]. The cities include Milwaukee, Chicago, St. Louis, Syracuse, Dayton, Pittsburgh, Denver, Columbus, Kansas City, Oklahoma City, Wichita, Memphis, Baltimore and Charleston. Each city is made up of tracts from the 2010 Census.

The Census tracts are pushed away from neighboring tracts based on the change in proportion of white and black populations between each neighboring tract.

Tracts having similar racial mix as their neighbors form groups. Spaces occur where there is a significant change in the racial makeup between neighboring tracts. The space is proportional to the change in racial composition between neighbors.

Fig. 5. Visualizing The Racial Divide: Chicago [12].

4. BENEFITS AND LIMITATIONS

D3 is an open source project, with it's source code readily available on Mike Bostock's github account [13]. It is free to use, which sets it apart from similar data visualization JavaScript libraries such as amCharts and FusionCharts, which need paid licenses. D3's development is supported by a big online community, where users can contribute to it's examples making it an exhaustive resource of various data visualizations.

D3's is also a drawing library unlike conventional data visualization tool-kits, that simply create data visualizations. D3 supports dynamic visualizations, as opposed to mere static visualizations.

D3 can be started using just one line of code, which is not the case with other data visualization tool-kits, often requiring a long installation procedure and periodic updating. D3's compatibility with web standards provide an added advantage that visualizations can be shared and viewed without the need for additional plug-ins. It is based on JavaScript which is compatible with browser's built-in debugger. This facilitates it's fast and easy debugging.

Amongst several advantages D3 has a few limitations. It's learning curve can be fairly steep. This is partly explained by the need for in depth knowledge of web standards, especially of SVG. Furthermore, the helper modules required for data transformations need studying too.

Unlike JavaScript libraries such as Google Charts, D3 does not offer in-built (standard) charts. This makes it less efficient as compared to other tool-kits with custom graphs.

5. USEFUL RESOURCES

The complete documentation of D3.js, including instructions to download and install it's latest release, examples, and several tutorials, in a systematic method are available on the main website of D3.js [3]. It is also a useful resource to D3's core concepts, techniques, blogs, books, courses, talks, videos and meetups [14].

The paper titled "D^3: Data-Driven Documents" authored by the creator of D3, compares D3 to existing web-based methods for visualization. The paper demonstrates how D3 is at least twice as fast as Protovis, through performance benchmarks. It also describes D3's potential for dynamic visualization [2].

Another website provides a complete path to create interactive visualization using D3.js [15]. It provides few real world examples and steps to create basic pie charts, animated bar charts and map.

6. CONCLUSION

The increasing popularity of JavaScript, has caused a major shift in the direction of web development with reduced dependencies on plug-ins. Consequently, developers are trying rely on the web browsers alone to avoid dependence on external plug-ins. This reduces the possibility of bugs or incompatibility issues as well the need to update. As an added benefit, developers can work with the existing web standards, increasing D3's compatibility with technologies like HTML, CSS and JavaScript. This makes preparation of data visualizations easy and readily available to everyone. D3.js is an toolkit for efficiently creating complex visualizations based on large datasets. It can be used for solving real world problems by creating visualisations to better infer trends in complex large data sets.

7. ACKNOWLEDGMENTS

This project was a part of the Big Data and Software and Projects (INFO-I524) course. I would like to thank Professor Gregor

von Laszewski and the associate instructors for their help and support during the course.

REFERENCES

[1] Mike Bostock, “Home,” Web Page, accessed: 2017-03-24. [Online]. Available: https://github.com/d3/d3/wiki

[2] M. Bostock, V. Ogievetsky, and J. Heer, “D3: Data-driven documents,” *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011. [Online]. Available: http://vis.stanford.edu/papers/d3

[3] Mike Bostock, “D3 Data Driven Documents,” Web Page, accessed: 2017-03-24. [Online]. Available: https://d3js.org/

[4] ——, “Gallery,” Web Page, accessed: 2017-03-24. [Online]. Available: https://github.com/d3/d3/wiki/Gallery

[5] Peter Cook, “Wimbledon 2013 Data Visualisations,” Web Page, accessed: 2017-03-24. [Online]. Available: http://charts.animateddata.co.uk/tennis/index.html

[6] Joseph, “Tennis-Data.co.uk,” Web Page, accessed: 2017-03-24. [Online]. Available: http://www.tennis-data.co.uk/wimbledon.php

[7] Peter Cook, “Wimbledon 2013 Circular Match Tree,” Web Page, accessed: 2017-03-24. [Online]. Available: http://charts.animateddata.co.uk/tennis/matchTree.html

[8] ——, “Wimbledon 2013 David Goliath,” Web Page, accessed: 2017-03-24. [Online]. Available: http://charts.animateddata.co.uk/tennis/davidgoliath.html

[9] ——, “Wimbledon 2013 Top 32 Players,” Web Page, accessed: 2017-03-24. [Online]. Available: http://charts.animateddata.co.uk/tennis/top32.html

[10] Pablo Navarro, “Earthquakes in Chile since 1900,” Web Page, accessed: 2017-03-24. [Online]. Available: http://pnavarrc.github.io/earthquake/

[11] Jim Vallandingham, “Visualizing The Racial Divide,” Web Page, accessed: 2017-03-24. [Online]. Available: http://vallandingham.me/racial_divide/

[12] ——, “Visualizing The Racial Divide,” Web Page, accessed: 2017-03-24. [Online]. Available: http://vallandingham.me/racial_divide/#ch

[13] Mike Bostock, “Mike Bostock,” Web Page, accessed: 2017-03-24. [Online]. Available: https://github.com/mbostock

[14] ——, “Tutorials,” Web Page, accessed: 2017-03-25. [Online]. Available: https://github.com/d3/d3/wiki/Tutorials

[15] Analytics Vidhya, “Newbie to D3.js Expert: Complete path to create interactive visualization using D3.js,” Web Page, accessed: 2017-03-24. [Online]. Available: https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/newbie-d3-js-expert-complete-path-create-interactive-visualization-d3-js/

Jupyter Notebook vs Apache Zeppelin - A comparative study

SRIRAM SITHARAMAN[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: srirsith@iu.edu*

March 26, 2017

With the development of new technologies for the purpose of exploring, analysing and visualizing big datasets, there exists a growing demand of a collaborative platform that combines the process of data analysis and visualization.The idea of computer notebooks has been around for a long time, starting with the early days of Matlab and Mathematica in the mid-to-late-80s. Two such platforms: Apache Zeppelin and Jupyter notebook are taken in to consideration for comparison. Both of them were ranked against characteristics such as their ability to support multiple programing techniques, multi-user support and integrated visualization.

Keywords: Jupyter, Zeppelin, Notebook, Comparison

https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf

CONTENTS

1. INTRODUCTION - JUPYTER NOTEBOOK

Jupyter notebook, part of project Jupyter is the third version of IPython notebook. It is a web based interactive development environment and supports multiple programming languages (Python, Julia, R etc.) [1]. It started supporting Julia, Python and R in its initial release, hence the term Jupyter. The web based interative environment provided by Jupyter is facilitated through a notebook interface which contains the programming code written by the user as well mark down elements and outputs that are generated as result of the written code[2]. Hence, Jupyter Notebook documents allows a seemless view of the code written and the corresponding output (graphs,tables,etc.). The Jupyter Notebook app is a client-server application as shown in the figure 1 that can be run on a local machine for personal use and can be accessed without the internet, or can be installed in a remote server which can be accessed via an internet connection. The final component of the Jupyter Notebook is the kernel which performs the execution of code written by user. Jupyter comes with a native Ipython kernel (supporting Python), and also supports 80+ programming languages' kernels [3].

2. INTRODUCTION - APACHE ZEPPELIN

Apache Zeppelin [5], similar to Jupyter notebook, is also a web based interactive notebook but aimed towards supporting big data and data analytics. It supports multiple programming languages. It is useful in exploration of data, creation of visualizations and sharing insights, as web pages, with various stakeholders involved in a project. It supports a range of programming languages like Scala, Spark SQL, Shell, etc with the default being scala. Zeppelin has an architecture similar to Jupyter notebook with an exception that notebook in Zeppelin supports the integration of multiple programming languages in a single notebook. Zeppelin's notebook is shipped with some basic charts which can be used to visualize output generated by any supported programming language. Apart from that, it has the option of pivot charts and Dynamic forms that can be

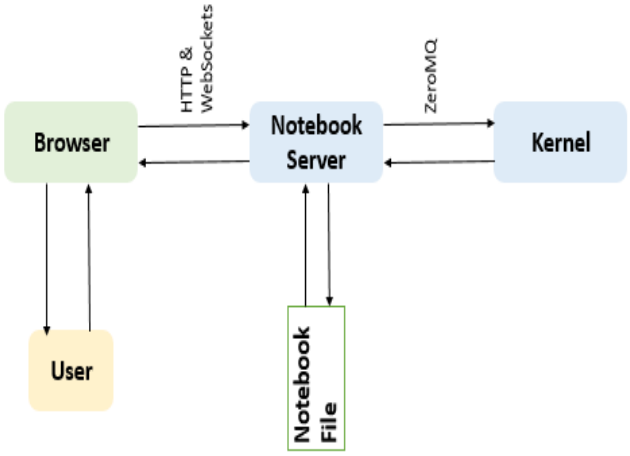


Fig. 1. Jupyter Architecture [4]

generated inside the notebook interface.

3. COMPARISON

This section would compare the following aspects of Apache Zeppelin and Jupyter notebook:

1. Interpreter configuration
2. Interface
3. Supported Languages
4. Visualization
5. Multi-user capability

3.1. Interpreter configuration

Zeppelin interpreter is a language/data-processing-backend that can be plugged into Zeppelin notebook. Currently, Zeppelin supports over 30 interpreters such as Scala (with Apache Spark), Python (with Apache Spark), Spark SQL, JDBC, Markdown, Shell, etc [6]. Zeppelin has a separate Interpreter configuration page that you can be accessed for multiple language parameters. For instance, Spark home directory as well as spark master string can be modified to our preference. It would create the Spark Context automatically so you don't need to deal with it in each notebook.For example, to use Scala code in Zeppelin, "%spark" has to be included to load the interpreter. Apache Zeppelin provides several interpreters as community managed interpreters [7] which can be installed at once if **netinst** binary package has been installed. It also supports installation of 3rd party interpreters.

For Jupyter, IPython is the default kernel defined as Kernel zero, which can be obtained through the kernel **ipykernel**. Jupyter supports the use of 80+ kernels and [3] shows how each of them can be installed, required dependencies and the correponding programming language version needed.

3.2. Interface

Zeppelin leverages a common UI framework with Bootstrap and Angular.js for the notebook interface. It is a easy to use interface with the support for creating dynamic forms within Zeppelin's notebook for which input can be obtained from a programming languages' output. Zeppelin provides an interface that is similar to RShiny's interactive web interface which allows user to manipulate in the front end rendered using Javascript with R running in the background [8].

Jupyter, on the other hand has a simple interface that is not user interactive as Zeppelin. When jupyter notebook is launched, the first page that is encountered is the Notebook Dashboard which shows the notebook files that has been created already and residing in the server. A new notebook can be created that is associated to a specific programming language's kernel (Python 2/Python 3/ R etc.) [9]. It's interface simple interface having a cell where the code can be written. Once the code is executed, the area below the cell would display the corresponding output.

3.3. Supported Languages

Zeppelin has the support for the 16 interpreters mentioned in Table 1. Since Apache Zeppelin has the default interpreter as spark which is a one of the major technology used for solving big data problems, the list includes interpreters like hbase, cassandra, elastic search etc. that works along with spark in solving big data problems.To overcome the limitation of these list of interpreters, Zeppelin provides the support for writing our own interpreter as described in [10].

Table 1. Programming languages supported by Apache Zeppelin

alluxio	file	jdbc	md
angular	flink	kylin	postgresql
cassandra	hbase	lens	python
elasticsearch	ignite	livy	shell

Jupyter, on the other hand has a huge list of about 80+ kernels being supported currently [3]. Though Jupyter has an upper-hand over Zeppelin over the number of supported programming languages, it lags behind Zeppelin in the support of using different programming language in the same notebook.

3.4. Visualization

Zeppelin provides the freedom to play around with various types of chart as shown in figure 2. It provides charting options by default for the output generated as a result of execution of code. The Apache Zeppelin community has been working on Project Helium [11], which aims to seed growth in all kinds of visualizations. This follows the model created by pluggable interpreters. Helium aims to make adding a new visualization simple, intuitive and can be accessed through a packaged code.

Jupyter, on the other hand has no charting options by default. Hence, it relies on existing charting libraries from the programming language that the notebook uses.

3.5. Multi-user capability

Zeppelin is still working towards providing multi user capability. Jupyter Hub [12] provides multiple user support for Jupyter. It simple to setup because it leverages the Linux users and groups to provide authentication.Three subsystems make up JupyterHub as shown in Figure 3. Each user cannot touch or see any other users because it's restricted at the OS layer. However,

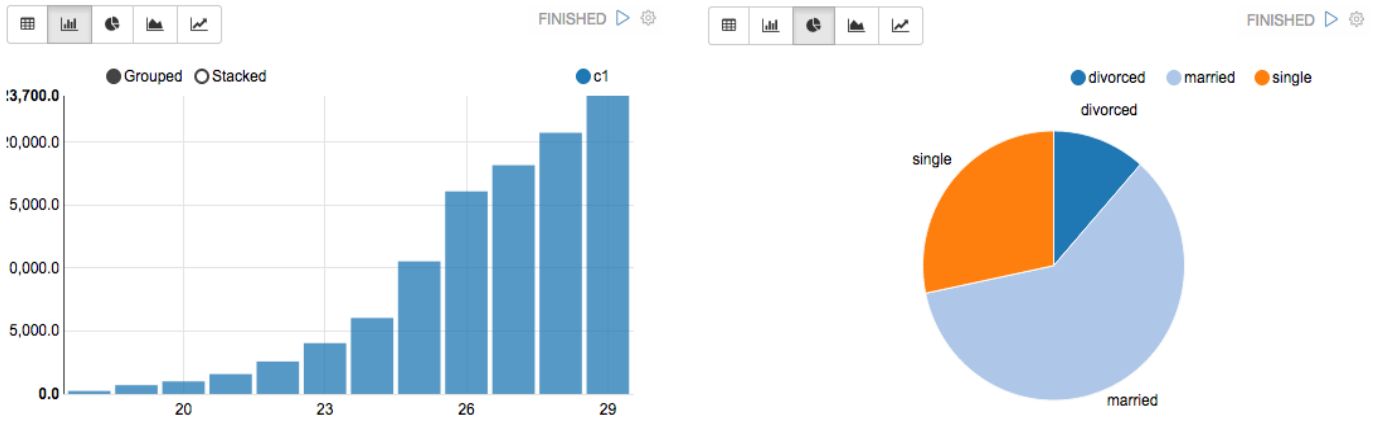


Fig. 2. Snapshot of Integrated graph - Apache Zeppelin [5]

Jupyter Hub has to be maintained in a single server which would provide access to multiple users.

1. a multi-user Hub (tornado process)
2. a configurable http proxy (node-http-proxy)
3. multiple single-user Jupyter notebook servers (Python/IPython/tornado)

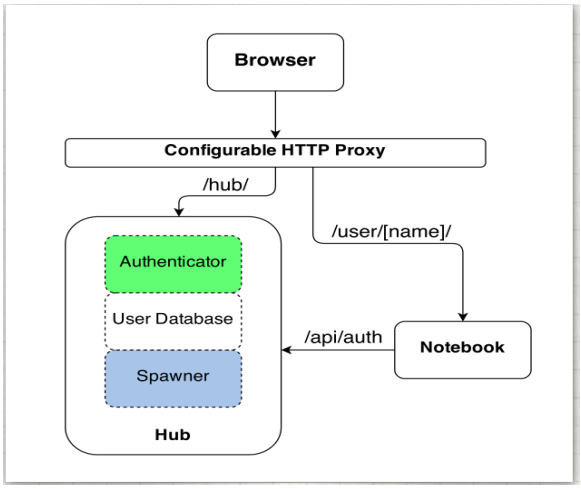


Fig. 3. Jupyter HUB Architecture [12]

3.6. Community support

Zeppelin is still in the incubation stage of Apache. It is progressing very slowly relatively to the status of Jupiter today. However, since Jupyter has a long history as IPython, there is lot of support for Jupyter in the online communtiy. A simple google search for the term **IPython** gives around 1.3 million results whereas for **Apache Zeppelin**, it is around 0.45 million. Apache Zeppelin is working with NFLabs, Twitter, Hortonworks, MapR, Pivotal, and IBM among many others delivering new features and fix issues in its platform [13].

4. ALTERNATIVES TO JUPYTER AND APACHE ZEPPELIN

4.1. Beaker Notebook

The Beaker Notebook is built on top of IPython kernel and was designed from the start to be a fully polyglot notebook [14]. It currently supports Python, Python3, R, Julia, JavaScript, SQL, Java, Clojure, HTML5, Node.js, C++, LaTeX, Ruby, Scala, Groovy, Kdb. It follows a cell type interface similar to Jupyter and Zeppelin and allows the user to use multiple programming languages across different cells in the same notebook. (i.e.) For example, the output of the code written in R in cell 1 can be accessed by a block of code written in Python in cell 2 for further manipulations.

4.2. SageMath

Sagemath is a is a free open-source mathematics software system [15] built for creating an open source alternative to Magma, Maple, Mathematica, and MATLAB. It is build on top of existing open-source packages: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R etc. It also offers a notebook type interface and the Sage Notebook recently moved to the cloud with SageMathCloud in collaboration with Google's cloud services.

5. CONCLUSION

The very need for experiments, explorations, and collaborations in scientific programming community is addressed by the evolution of these notebooks.Considering these into account, Apache Zeppelin is gaining upper hand over Jupyter in providing a fluid environment to solve big data problems apart from it being in the initial stages of defining multi-user support and relatively small community. With Zeppelin being a part of Apache community, it would be understandable that there would be constant growth and updates. This can be strengthened from the fact that Apache Zeppelin is in the initial stages of developing Helium which would make visualization easy to use for big data problems.

REFERENCES

[1] Wikipedia, "Jupyter -debian wiki," dec 2016, [Online; accessed 23-March-2017]. [Online]. Available: https://wiki.debian.org/Jupyter

[2] Jupyter, “What is the jupyter notebook?” Web Page, jan 2017, accessed: 2017-03-02. [Online]. Available: http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html
[3] ——, “Jupyter kernels,” Web Page, feb 2017, accessed: 2017-03-02. [Online]. Available: https://github.com/jupyter/jupyter/wiki/Jupyter-kernels
[4] ——, “How ipython and jupyter notebook work,” Web Page, feb 2017, accessed: 2017-03-02. [Online]. Available: http://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html
[5] Apache, “Apache zeppelin,” Web Page, nov 2016, accessed: 2017-03-02. [Online]. Available: http://zeppelin.apache.org/
[6] ——, “Interpreters in apache zeppelin,” Web Page, nov 2016, accessed: 2017-03-02. [Online]. Available: http://zeppelin.apache.org/docs/latest/manual/interpreters.html
[7] ——, “Interpreters installation,” Web Page, nov 2016, accessed: 2017-03-02. [Online]. Available: https://zeppelin.apache.org/docs/0.6.0/manual/interpreterinstallation.html
[8] RStudio, “Shiny,” Web Page, dec 2016, accessed: 2017-03-22. [Online]. Available: https://shiny.rstudio.com/
[9] Jupyter, “Ui components - jupyter notebook,” Web Page, feb 2017, accessed: 2017-03-02. [Online]. Available: http://jupyter-notebook.readthedocs.io/en/latest/ui_components.html
[10] Apache, “Writing a new interpreter,” Web Page, nov 2016, accessed: 2017-03-02. [Online]. Available: http://zeppelin.apache.org/docs/latest/development/writingzeppelininterpreter.html#make-your-own-interpreter
[11] Wikipedia, “Helium proposal,” mar 2016, [Online; accessed 21-March-2017]. [Online]. Available: https://cwiki.apache.org/confluence/display/ZEPPELIN/Helium+proposal
[12] Jupyter, “Jupyterhub,” Web Page, feb 2017, accessed: 2017-03-22. [Online]. Available: https://jupyterhub.readthedocs.io/en/latest/
[13] hortonworks, “Jupyterhub,” Web Page, jun 2016, accessed: 2017-03-23. [Online]. Available: https://hortonworks.com/blog/apache-zeppelin-road-ahead/
[14] T. Sigma, “Beaker notebook,” Web Page, dec 2016, accessed: 2017-03-25. [Online]. Available: http://beakernotebook.com/
[15] S. Foundation, “Beaker notebook,” Web Page, jan 2017, accessed: 2017-03-25. [Online]. Available: http://www.sagemath.org/

Google BigQuery - A data warehouse for large-scale data analytics

SAGAR VORA[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*

March 27, 2017

There has been an increase in the volume of relational data generated today through a large number of sources. The large volume of data forces us to find solutions which can cope with them. Recently several hybrid approaches like HadoopDB, Hive, etc have been introduced to handle this large data. Although these have been successful in handling large data, but their architecture makes them inefficient to handle suboptimal execution strategies. Moreover this data is the information that companies would like to explore and analyse quickly to identify strategic answers to business. Therefore, in order to solve the problem of traditional database management systems to support large volumes of data arises Google's BigQuery platform. This solution runs in cloud, SQL-like queries against massive quantities of data, providing real-time insights about the data. In this paper, we will analyze the main features of BigQuery that Google offers to manage large-scale data along with its comparison with other data storage platforms.

Keywords: BigQuery, BigData, Google, Cloud, Database, IaaS, PaaS, SaaS, SQL

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IR-2041/report.pdf

INTRODUCTION

Nowadays, the amount of data being collected, stored and processed continues to grow rapidly. Therefore high performance and scalability have become two essential requirements for data analytics systems. Querying massive datasets can be time consuming and expensive without the right hardware and infrastructure. BigQuery solves this problem by enabling super-fast, SQL-like queries against append-only tables, using the processing power of Google's infrastructure. Google BigQuery [1] [2] is a cloud web service data warehouse used for large-scale data processing. It is ideal for businesses that cannot invest in infrastructure to process a huge amount of information. This platform allows to store and retrieve large amounts of information in near real time with main focus on data analysis.

CLOUD COMPUTING

Cloud computing [3] [4] allows access to computing resources easily scalable and virtualized via the Internet. The use becomes simpler because users need not to have knowledge, experience or management of the infrastructure. There are usually three types of cloud offerings:

- Infrastructure as a service (IaaS) which provides basic compute and storage resources including servers, networking,etc.
- Platform as a service (PaaS) which provides a platform allowing users to develop, run and manage applications without the complexity of building and maintaining the infrastructure.
- Software as a Service (SaaS) which refers to one of the most known cloud services, which consists of applications running directly in the cloud provider.

GOOGLE BIGQUERY

To solve the problems faced by Hadoop[5] MapReduce[6], Google developed Dremel application in order to process large volumes of data. Dremel was designed to deliver high performance on data which was distributed across multiple servers and SQL support. But in 2012 at Google I/O event, it was announced that they would no longer support Dremel and this led to the beginning of BigQuery which became then the high-performance cloud offering of google.
Google BigQuery [2] platform is a SaaS as a model in the cloud. It is not a reporting system and does not have an interface that allows the operation of the information but it is ideal to export results by Tableau, QlikView, Excel, among others including the tools of Business Intelligence (BI) as seen in figure 1. Projects are top-level containers that store information about billing and authorized users, and they contain BigQuery data. When we

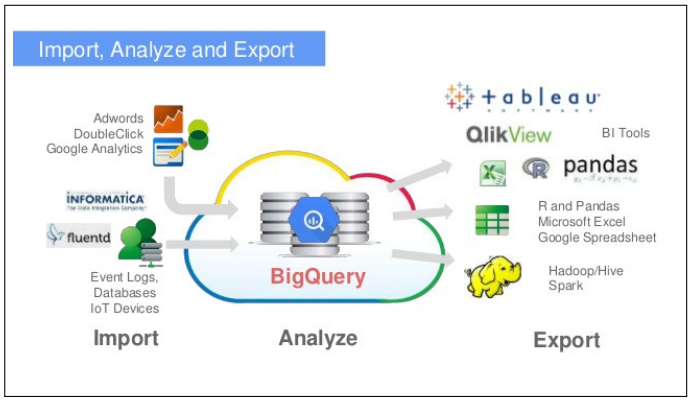


Fig. 1. [7] System Architecture of BigQuery

create a new project, it is identified by a name, authorized users and data[8].
Data which is generated from a variety of sources like event logs, relational databases, IoT devices like sensors, social media websites, etc is given as an input to BigQuery which processes it to generate some meaningful analysis according to the requirement. The final data could be represented and exported using Tableau, Qlikview and other BI tools. It can also be exported on Hadoop system for parallel processing as in figure 1.

FEATURES OF BIGQUERY

Google BigQuery[1] presents some characteristics like velocity, scalability, simplicity, security and multiple access methods.

Velocity

BigQuery[1] can process millions of information which is not indexed, in seconds due to columnar storage, and tree architecture.

Columnar Storage

The data instead of being stored in terms of rows, is stored as columns and thus storage will be oriented. This will result in scanning of only the required values, largely reducing latency. This storage also allows for a higher compression ratio. Google reports [9] that they can achieve columnar compression ratios of 1:10 as opposed to 1:3 when storing data in a traditional row-based format. The reason behind this is the similarity of data stored in the columns as the variation is significantly lower than when the data is stored in rows.

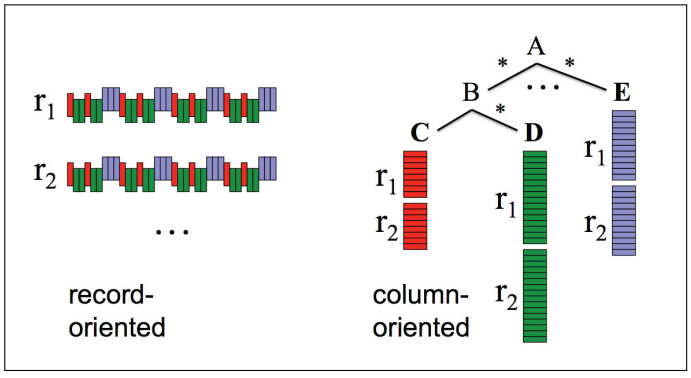


Fig. 2. [9] Row-oriented vs Column-oriented Storage

Tree Architecture

This is used for processing queries and aggregating results across different nodes. BigQuery is spread across thousands of servers. The data is shared across multiple machines as in figure 2. This helps retrieve data much faster. Let us see this with the help of 2 examples[9].
Here, A is our root server, with B being an intermediate server and C, D, E being leaf servers having local disk storage.

Example 1: Speed
Statement: List out all the customer names starting with 'G'.
Let us assume Node C contains customer names. Hence, it is as simple as traversing A -> B -> C and looking at the datasets Cr1 and Cr2 for names starting with G. One need not look at A -> B -> D and A -> E. Hence, in this simple scenario, we have already achieved a search time of 0.33x that of a typical RDBMS (assuming equal column sizes).

Example 2: Parallel Processing
Statement: Count all the customer names starting with 'G'.

- Root A passes the query to intermediate B.
- B translates the query to Sum (Count all the customer names starting with 'G').
- Intermediate B passes this query to Leaf C.
- Leaf C has tablets (Horizontal shared tables) r1 and r2.
- C accesses these tablets in parallel, runs quick counts and passes the count of Cr1 and Cr2 to B.
- B sums Cr1 and Cr2 and passes it to the root A for output.

Now, if this architecture was scaled to thousands of leaf servers and hundreds of intermediate servers. This achieves enormous amounts of parallel processing by utilizing a small percentage of the CPU processing and disk I/O of each server as opposed to 100% usage of fewer servers.

Scalability

It is the ability to manage huge data size with millions of records reaching terabytes of information, without space limits[1].

Simplicity

BigQuery provides a simple interface to upload and execute browse through a query language similar to SQL as shown in figure 3.

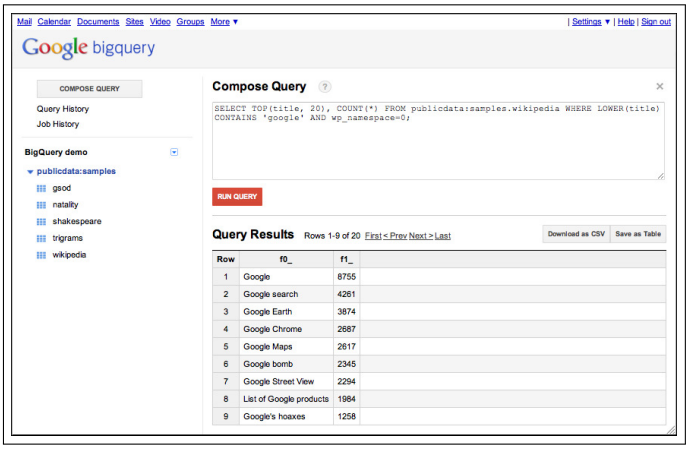


Fig. 3. [10] Big Query Sample User Interface

Multiple permissions

It is the capacity to manage different access permissions, read-only, editing or owner.

Security

To ensure security, the solution makes use of SSL (Secure Sockets Layer)

Multiple access methods

We can access the service in different ways. We can use a BigQuery browser tool, a bq command-line tool or a REST-based API.

- BigQuery Browser Tool: With this tool it is possible to easily browse, create tables, run queries, and export data to Google Cloud Storage.
- bq command-line Tool: This Python command - line tool permits to manage and query the data.
- REST API: We can access BigQuery making calls to the REST API using a variety of client libraries such as Java, PHP or Python.

EXPERIMENT

In [11], the performance of BigQuery is compared against Impala, Shark, Hive, Redshift and Tez by using following benchmark. The benchmark allows 3 different sizes of datasets - tiny, 1node and 5nodes. The largest dataset is 5nodes which has 'rankings' table with 90 million records and 'uservisits' table with 775 million records. The data is generated using Intel's Hadoop benchmark tools. The data itself are available at s3://big-data-benchmark/pavlo/[text | text-deflate | sequence | sequence-snappy]/[suffix]. The two tables have the following schemas:

Ranking table schema: (lists websites and their page rank)
- pageURL (String)
- pageRank (Integer)
- avgDuration (Integer)

Uservisits table schema: (Stores server logs for each web page)
- sourceIP (String)
- destURL (String)
- visitDate (String)
- adRevenue (Float)
- userAgent (String)
- countryCode (String)
- languageCode (String)
- searchWord (String)
- duration (Integer)

The benchmark measures response time on a handful of relational queries: scans, aggregations, and joins. Some of the query results are larger than 128MB.

- Scan Query

  This query has 3 permutations, when X is 1000, X is 100 and then X is 10:

  SELECT pageURL, pageRank FROM [benchmark.rankings] WHERE pageRank > X

- Aggregation Query

  This query has 3 permutations, when X is 8, X is 10 and then X is 12:

  SELECT SUBSTR(sourceIP, 1, X) AS srcIP, SUM(adRevenue) FROM [benchmark.uservisits] GROUP EACH BY srcIP

- Join Query

  Like other queries earlier, this one also has 3 permutations: X is '1980-04-01', X is '1983-01-01' and X is '2010-01-01':

  SELECT sourceIP, sum(adRevenue) AS totalRevenue, avg(pageRank) AS pageRank FROM [benchmark.rankings] R JOIN EACH(SELECT sourceIP, destURL, adRevenue FROM [benchmark.uservisits] UV WHERE UV.visitDate > "1980-01-01" AND UV.visitDate < X) NUV ON (R.pageURL = NUV.destURL) GROUP EACH BY sourceIP ORDER BY totalRevenue DESC LIMIT 1

	Query 1A	Query 1B	Query 1C
Redshift	2.49	2.61	9.46
Impala (Disk)	12.015	12.015	37.085
Impala (Mem)	2.17	3.01	36.04
Shark (Disk)	6.6	7	22.4
Shark (Mem)	1.7	1.8	3.6
Hive	50.49	59.93	43.34
Tez	28.22	36.35	26.44
BigQuery	4.6	14.6	11.4

	Query 2A	Query 2B	Query 2C
Redshift	25.46	56.51	79.15
Impala (Disk)	113.72	155.31	277.53
Impala (Mem)	84.35	134.82	261.015
Shark (Disk)	151.4	164.3	196.5
Shark (Mem)	83.7	100.1	132.6
Hive	730.62	764.95	833.3
Tez	377.48	438.03	427.56
BigQuery	15.1	24.4	11.4

	Query 3A	Query 3B	Query 3C
Redshift	33.29	46.08	168.25
Impala (Disk)	108.68	129.815	431.26
Impala (Mem)	41.21	76.005	386.6
Shark (Disk)	111.7	135.6	382.6
Shark (Mem)	44.7	67.3	318
Hive	561.14	717.56	2374.17
Tez	323.06	402.33	1361.9
BigQuery	9.3	9.1	11.2

Fig. 4. [11] Comparison of BigQuery with other storage platforms

Figure 4 shows the experimental results. For this experiment, each query was executed 10 times and the results are median response time in seconds. From 4, it shows that only in Query A BigQuery took more time than others but while executing Query 2 and 3, it executed them faster than other platforms. This shows that BigQuery can produce amazing results when processing complex queries on large datasets.

CONCLUSION

Google BigQuery is a cloud-based database service that is able to process large data sets quickly. BigQuery allows to run SQL-like queries against multiple terabytes of data in a matter of seconds. It is suitable for ad-hoc OLAP/BI queries that require results as fast as possible. As a cloud-powered massively parallel query database it provides extremely high full-scan query performance and cost effectiveness compared to traditional data warehouse solutions and appliances.

ACKNOWLEDGEMENTS

I would like to thank my professor Gregor von Laszewski and all the associate instructors for their constant technical support.

REFERENCES

[1] "What is bigquery," Web Page, accessed: 2017-3-24. [Online]. Available: https://cloud.google.com/bigquery/

[2] S. Fernandes and J. Bernardino, "What is bigquery?" in *Proceedings of the 19th International Database Engineering & Applications Symposium*. New York, NY, USA: ACM, 2014, pp. 202–203. [Online]. Available: http://doi.acm.org.proxyiub.uits.iu.edu/10.1145/2790755.2790797

[3] "What is cloud computing," Web Page, accessed: 2017-3-24. [Online]. Available: https://apprenda.com/library/cloud

[4] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proceedings of the Second International Workshop on Testing Database Systems*. New York, NY, USA: ACM, 2009, pp. 9:1–9:6. [Online]. Available: http://doi.acm.org/10.1145/1594156.1594168

[5] "Apache hadoop," Web Page, accessed: 2017-3-25. [Online]. Available: http://hadoop.apache.org/

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, published as an Article. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[7] K. Sato, "Fluentd + google bigquery," Web Page, Mar. 2014, accessed: 2017-3-24. [Online]. Available: https://www.slideshare.net/GoogleCloudPlatformJP/google-for-1600-kpi-fluentd-google-big-query

[8] "What is bigquery? bigquery documentation google cloud platform," Web Page, accessed: 2017-3-23. [Online]. Available: https://cloud.google.com/bigquery/what-is-bigquery

[9] V. Agrawal, "Google bigquery vs mapreduce vs powerdrill," Web Page, accessed: 2017-3-23. [Online]. Available: http://geeksmirage.com/google-bigquery-vs-mapreduce-vs-powerdrill

[10] J.-k. Kwek, "Google bigquery service: Big data analytics at google speed," Blog, Nov. 2011, accessed: 2017-3-23. [Online]. Available: https://cloud.googleblog.com/2011/11/google-bigquery-service-big-data.html

[11] V. Solovey, "Google bigquery benchmark," Blog, Jun. 2015, accessed: 2017-3-23. [Online]. Available: https://www.doit-intl.com/blog/2015/6/9/bigquery-benchmark

Hive

DIKSHA YADAV[1,*,+]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: yadavd@umail.iu.edu*
[+] *HID - S17-IR-2044*

Paper-002, March 27, 2017

Hive is an open source data warehousing solution which is built on top of Hadoop. It structures data into understandable and conventional database terms like tables, columns, rows and partitions. It supports HiveQL queries which have structure like SQL queries. HiveQL queries are compiled to map reduce jobs which are then executed by Hadoop. Hive also contains Metastore which includes schemas and statistics which is useful in query compilation, optimization and data exploration.

Keywords: Hive, Hadoop, HiveQL, SQL

https://github.com/diksha2112/sp17-i524/tree/master/paper2/S17-IR-2044/report.pdf

INTRODUCTION

The reason behind development of Hive is making it easier for end users to use Hadoop. Map reduce programs were required to be developed by users for simple to complex tasks. It lacked expressiveness like query language. So, it was a time consuming and difficult task for end users to use Hadoop. For solving this problem Hive was built in January 2007 and open sourced in August 2008. Hive is an open source data warehousing solution which is built on top of Hadoop. It structures data into understandable and conventional database terms like tables, columns, rows and partitions. It supports HiveQL queries which have structure like SQL queries. HiveQL queries are compiled to map reduce jobs which are then executed by Hadoop. Hive also contains Metastore which includes schemas and statistics which is useful in query compilation, optimization and data exploration[1]

ARCHITECTURE

Hive architecture includes, Database-It consists of tables created by the user. Metastore-It contains information about the system. It can be accessed by different components as and when needed. Interfaces-User interface and Application programming interface both are present in hive. Driver-manage HiveQL statements at every stage. Query compiler-It compiles HiveQL queries to acyclic graphs (directed) representing map reduce tasks. Execution Engine- It executes the tasks generated by the compiler. Hive Server- It provide JDBC/ODBC server and thrift interface[2]

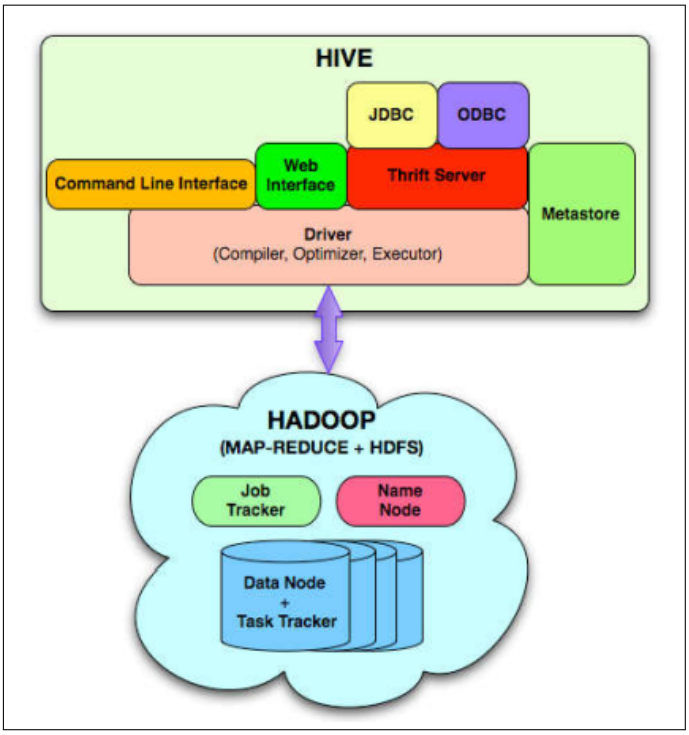


Fig. 1. Hive Architecture

HIVEQL QUERY FORMAT

SELECT [ALL | DISTINCT] select-expr, select-expr, ...
FROM table-reference
[WHERE where-condition]
[GROUP BY col-list]
[HAVING having-condition]
[CLUSTER BY col-list | [DISTRIBUTE BY col-list] [SORT BY col-list]]
[LIMIT number]
[3]

SYSTEM REQUIREMENTS

Hive is cross platform. So, It does not need any specific operating system to work.

COMPARISON OF HIVE WITH OTHER TRADITIONAL DATABASES

Traditional databases follow schema on write approach while Hive follows schema on read approach. In schema on write, databases checks at load time if the data follows the table representation given by user while in schema on read approach, it is checked at run time only. This saves the time for hive to load the data when traditional databases takes longer time[4]

POPULARITY OF HIVE

The popularity of hive increasing with time. This can be proved by the following plot made by DB Engines Ranking. It ranks database management systems according to their status and popularity. Following plot shows popularity of hive with time.

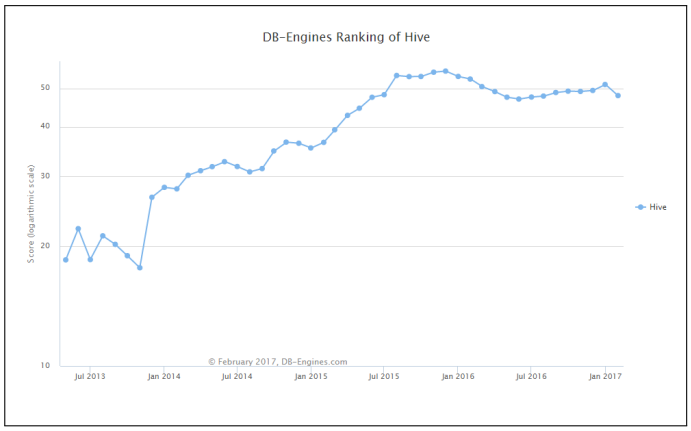


Fig. 2. Hive Popularity

[5]

RESOURCES FOR LEARNING HIVE

Someone new to hive can start learning it by going through the following links in sequence: Install Hivehttps://www.edureka.co/blog/apache-hive-installation-on-ubuntu?utm_source=quora&utm_medium=crosspost&utm_campaign=social-media-edureka-ab
Hive Tutorialhttps://www.edureka.co/blog/hive-tutorial/?utm_source=quora&utm_medium=crosspost&utm_campaign=social-media-edureka-ab
Top Hive commands with exampleshttps://www.edureka.co/blog/hive-commands-with-examples?utm_source=quora&utm_medium=crosspost&utm_campaign=social-media-edureka-ab

ACKNOWLEDGEMENT

I am also grateful to Dr. Gregor von Laszewski for providing the appropriate paper template.

CONCLUSION

Since Hive is making the use of Hadoop easier for users, its popularity is increasing with time.

REFERENCES

[1] "Hive," Web Page, online; accessed 24-March-2017. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Hive
[2] "Hive article," Web Page, online; accessed 24-March-2017. [Online]. Available: https://docs.treasuredata.com/articles/hive
[3] A. T. J. S. N. J. Z. S. P. C. S. A. H. L. P. W. R. Murthy, "Hive-a warehousing solution over map reduce framework," Paper, online; accessed 23-March-2017. [Online]. Available: http://laser.inf.ethz.ch/2013/material/breitman/additionalpercent20reading/hive.pdf
[4] ——, "Hive-a petabyte scale datawarehouse using hadoop," Paper. [Online]. Available: http://infolab.stanford.edu/~ragho/hive-icde2010.pdf
[5] "Ranking hive," Web Page, online; accessed 20-March-2017. [Online]. Available: http://db-engines.com/en/ranking_trend/system/Hive