Big Data Technologies

Editor: Gregor von Laszewski

March 21, 2017

0.1 Contributors

Name	HID	Title	Pages
Avadhoot Agasti	SL-IO-3000	TBD	1
Abhishek Gupta	S17-IO-3005	TBD	1
Matthew Lawson	S17-IO-3010	TBD	1

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 10, 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 19, 2017

Amazon Kinesis [1] provides a software-as-a-service(SAAS) platform to application developers on Amazon Web Services(AWS) platform, which 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] gives application developers collect and analyze streaming data in real time. The stream 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 changes. 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 streams nodes go 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 type of processing engines

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

Kinesis streams can used to process incoming data processing from multiple datasources while 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 workers(or shards) using the incoming partition key.

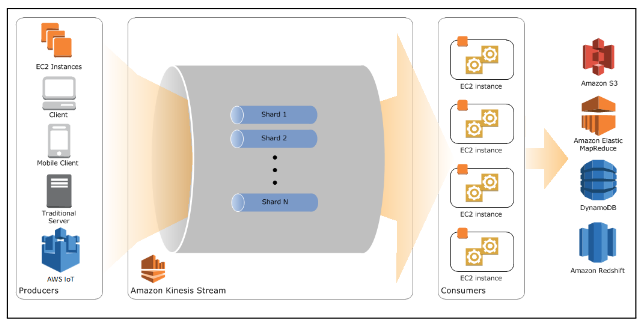

Fig. 1. Kinesis streams building blocks [2]

Following are key components in streams architecture [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 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,

```
record 0 --|
record 1   |         [ Aggregation ]
    ...    |--> Amazon Kinesis record 0 --|
    ...    |                              |
record A --|                              |
                                          |
    ...                 ...               |
                                          |
record K --|                              |
record L   |                              |      [ Collection ]
    ...    |--> Amazon Kinesis record C --|--> PutRecords Request
    ...    |                              |
record S --|                              |
                                          |
    ...                 ...               |
                                          |
record AA--|                              |
record BB  |                              |
    ...    |--> Amazon Kinesis record M --|
    ...    |
record ZZ--|
```

Fig. 2. Aggregation of records

2.3. Consumer

Consumers subscribe to one or more streams. Consumer can be one of the application running on aws or hosted on EC2.

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 values 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 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 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 reads 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 which has limit of 50 shards. Each shard can support up to 5 transactions per second for reads and 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 workflow 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

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

- 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] an 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 you demand increases or decreases and pay based on your usage. Only downside of Kinesis it that it cannot run on a private or hybrid cloud rather can only 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. Its comparable with 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 19, 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

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" [2]. 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. [3]

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* [4]. 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 [5]. 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 [1]

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* [6]. 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 [7]. 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 [4]. 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 [4].

ROS addresses this issue with its *transform* library. The *tf* library tracks components of a robot using three-dimensional coordinate frames [8]. 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 [4]. 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 [9]." ROS' diagnostics take advantage of the aforementioned message system

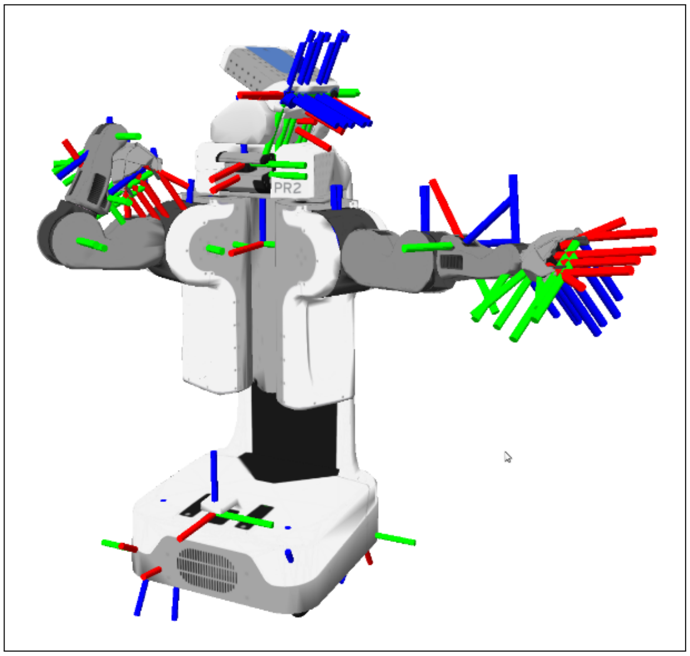

Fig. 2. A Simulated Robot with Many Coordinate Frames [8]

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 [10]. 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 [5]; 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* [4], 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 [11], 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 [12].

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 [13].

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 [14].

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* [15], 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 [12].

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. [16]" 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 [17].

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.

REFERENCES

[1] H. Boyer, "Open Source Robotics Foundation And The Robotics Fast Track," web page, nov 2015, accessed 19-mar-2017. [Online]. Available: https://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/11/rft-boyer.pdf

[2] Open Source Robotics Foundation, "About ROS," Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: http://www.ros.org/about-ros/

[3] National Instruments, "A Layered Approach to Designing Robot Software," Web page, mar 2017, accessed 18-mar-2017. [Online]. Available: http://www.ni.com/white-paper/13929/en/

[4] Open Source Robotics Foundation, "Why ROS?: Features: Core components," Web page, mar 2017, accessed 17-mar-2017. [Online]. Available: http://www.ros.org/core-components/

[5] ——, "ROS graph concepts: Messages," Web page, aug 2016, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/Messages

[6] ——, "Services," web page, feb 2012, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/Services

[7] ——, "actionlib: Package summary," Web page, feb 2017, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/actionlib

[8] ——, "tf2: Package summary," Web page, mar 2016, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/tf2

[9] ——, "diagnostics: Package Summary," Web page, aug 2015, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/diagnostics

[10] ——, "ROS command-line Tools," Web page, aug 2015, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/ROS/CommandLineTools

[11] ——, "Why ROS?: Is ROS for me?" Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: www.ros.org/is-ros-for-me/

[12] ——, "Robots Using ROS," Web page, mar 2017, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/Robots

[13] Open Source Research Foundation, "APIs," Web page, mar 2016, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/APIs

[14] Wikipedia, "BSD licenses — Wikpedia, the free encyclopedia," Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/BSD_licenses

[15] ——, "Free and open-source software – Wikipedia, the free encyclopedia," Web page, mar 2017, accessed 18-mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Free_and_open-source_software

[16] Fetch Robotics, Inc., "Automated Data Collection Platform," Web page, mar 2017, accessed 19-mar-2017. [Online]. Available: http://fetchrobotics.com/automated-data-collection-platform/

[17] Retail Touch Points, "Trax and Fetch Robotics Unite for Better Retail," Web page, jan 2017, accessed 19-jan-2017. [Online]. Available: http://www.retailtouchpoints.com/features/news-briefs/trax-and-fetch-robotics-unite-for-better-retail-insights

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.

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.