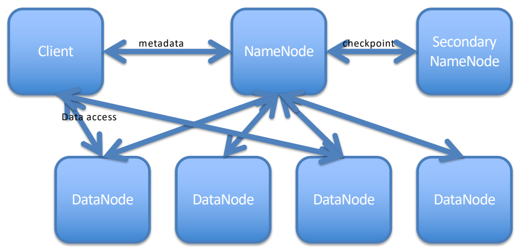
Big Data Engineering

Other Tools and Libraries

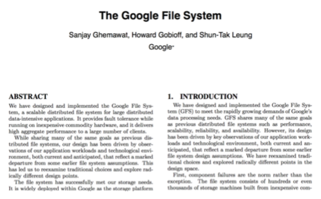
Julie Weeds
March 2019

HDFS in a nutshell



HDFS inspiration

- Google File System
  - Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). ACM, New York, NY, USA, 29-43.



The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive com-

HDFS overview

- Good for streaming access to large files, reliability, scale
- Not good for random access, small files
- Blocks of data 64Mb in size (configurable)
- Each block can be replicated across multiple data nodes for High Availability (HA)

HDFS Usage

- Spotify has 1600+ nodes, storing 60+ petabytes of data
  - https://www.usenix.org/system/files/conference/fast17/fast17-niazi.pdf
- One of Facebook's largest clusters (based on HDFS) holds more than
  100 PB of data, processing more than 60,000 Hive queries a day
  - https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/

HopFS

- HopFS is a drop-in replacement for HDFS, based on HDFS v2.0.4.

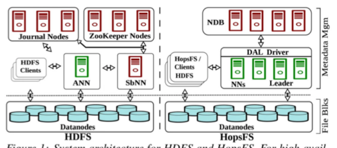


Figure 1: System architecture for HDFS and HopsFS. For high availability, HDFS requires an Active NameNode (ANN), at least one Standby NameNode (SbNN), at least three Journal Nodes for quorum-based replication of the write ahead log of metadata changes, and at least three ZooKeeper instances for quorum based coordination. HopsFS supports multiple stateless namenodes that access the metadata stored in NDB database nodes.

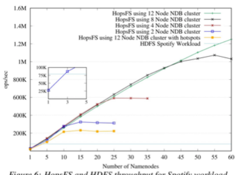
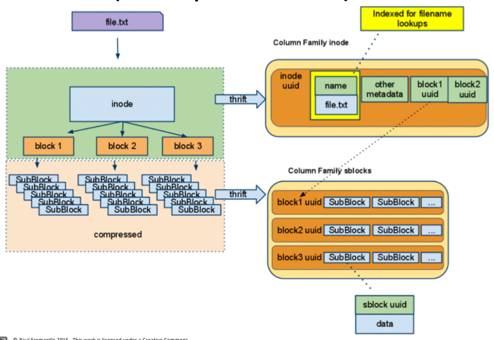


Figure 6: HopsFS and HDFS throughput for Spotify workload.

CassandraFS (not open source)



Amazon S3

- Simple Storage Service
- Unlimited storage of files
  - Up to 5 terabytes each
  - Stored in named "buckets"
  - Accessible via AWS APIs or HTTP
  - Authenticated or Public

Spark packages

- A wide set of plugins
  - Currently 148 community donated plugins
- Data connectors
  - Cassandra, Couchbase, Mongo, CSV, etc
- Machine Learning, Neural networks
- Streaming
- etc

Using Spark Packages

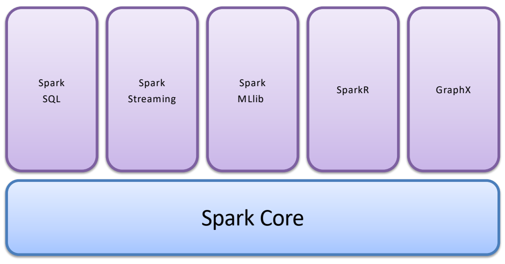
Automatic download from the web:

```
bin/spark-shell
    --packages com.databricks:spark-csv_2.11:1.2.0
```

Locality

- Spark understands the locality of data:
  - Already in memory
  - HDFS location
  - Cassandra location

Spark Extras



Spark Extras

- Spark Streaming
  - Realtime analysis in Spark
- Spark MLLib
  - Like Mahout – Machine learning in Spark
- GraphX
  - Graph processing in Spark
- SparkR
  - R statistical analysis on Spark

Spark MLlib

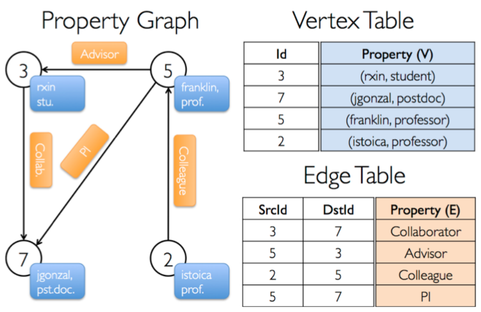
- Simple stats and correlation testing
- Classification and regression
- Collaborative Filtering
  - Alternating Least Squares
- Clustering
  - k-means, etc
- Frequent Pattern Mining
- Plus more

MLlib example

```
from pyspark.mllib.fpm import FPGrowth

data = sc.textFile("data/mllib/sample_fpgrowth.txt")

transactions = data.map(lambda line: line.strip().split(' '))

model = FPGrowth.train(transactions, minSupport=0.2,
    numPartitions=10)

result = model.freqItemsets().collect()
for fi in result:
    print(fi)
```

GraphX

-



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

R

- R is an open source system for statistics and graphics
  - Based on the S language from AT&T Bell Labs
- Supports a wide variety of statistical techniques and graphing tools
- An extensible set of packages that provide extra functions via CRAN
  - The Comprehensive R Archive Network

SparkR

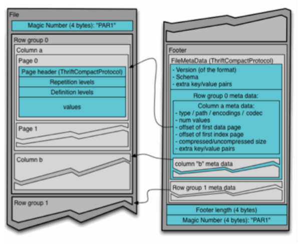
- A lightweight approach to use Spark from within R
- Also works with MLlib for machine learning
- Allows complex statistical analysis to be done on a Spark cluster

Apache Avro

- A compact data storage and transmission system
  - Uses schemas of data to ensure it can be read by the receiver
  - Supports dynamic typing
- Used by RPC or data collection systems
  - Fast binary protocols
- Also supports storage
  - Hence used by many Big Data apps including Hadoop and Spark

Apache Parquet

- Apache Parquet is a columnar data storage model
  - Works with Hadoop, Spark and many others
  - Efficient storage of data
  - Based on another Google system called Dremel


File
Magic Number (4 bytes): "PAR1"
Row group 0
Column a
Page 0
Page header (ThriftCompactProtocol)
Repetition levels
Definition levels
values
Page 1
Column b
Row group 1
Footer
FileMetaData (ThriftCompactProtocol)
- Version (of the format)
- Schema
- extra key/value pairs
Row group 0 meta data:
Column a meta data:
- type / path / encodings / codec
- num values
- offset of first data page
- offset of first index page
- compressed/uncompressed size
- extra key/value pairs
column "b" meta data
Row group 1 meta data
Footer length (4 bytes)
Magic Number (4 bytes): "PAR1"

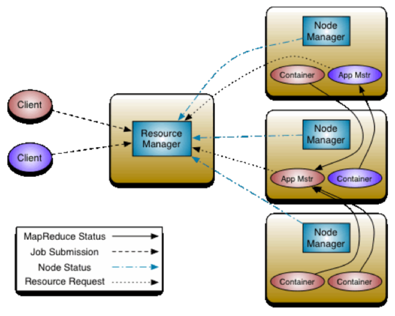
Cluster management systems for Big Data

- YARN
  - Part of Hadoop but significantly rebuilt since Hadoop 1
- Mesos
  - Popular Apache project
  - Built to be a resource manager for a complete datacenter
    - Supports many workloads (e.g. Docker as well as Spark)

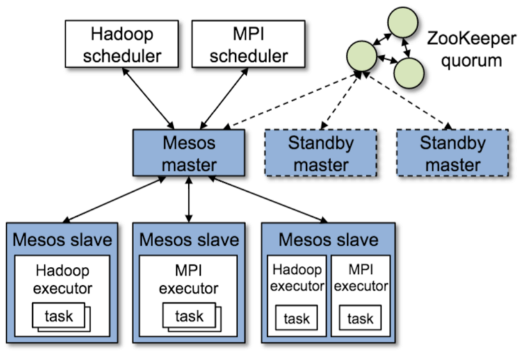
What is YARN?

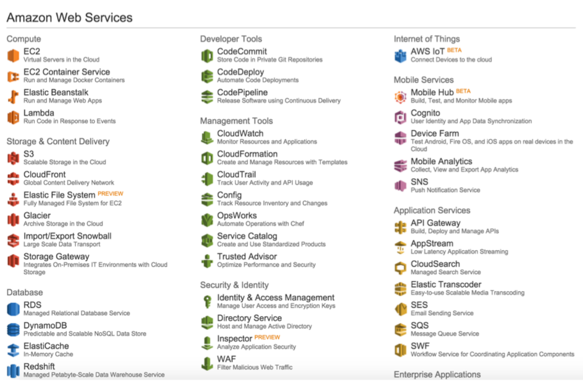
- **YARN is the system that runs your code on multiple nodes**
- Hadoop 2.0 replacement for the cluster manager
  - Basically a model to distribute and manage workloads
  - Not just MapReduce but supports other workloads

YARN architecture



Apache Mesos





Amazon Web Services

Compute
- EC2 — Virtual Servers in the Cloud
- EC2 Container Service — Run and Manage Docker Containers
- Elastic Beanstalk — Run and Manage Web Apps
- Lambda — Run Code in Response to Events

Storage & Content Delivery
- S3 — Scalable Storage in the Cloud
- CloudFront — Global Content Delivery Network
- Elastic File System PREVIEW — Fully Managed File System for EC2
- Glacier — Archive Storage in the Cloud
- Import/Export Snowball — Large Scale Data Transport
- Storage Gateway — Integrates On-Premises IT Environments with Cloud Storage

Database
- RDS — Managed Relational Database Service
- DynamoDB — Predictable and Scalable NoSQL Data Store
- ElastiCache — In-Memory Cache
- Redshift — Managed Petabyte-Scale Data Warehouse Service

Developer Tools
- CodeCommit — Store Code in Private Git Repositories
- CodeDeploy — Automate Code Deployments
- CodePipeline — Release Software using Continuous Delivery

Management Tools
- CloudWatch — Monitor Resources and Applications
- CloudFormation — Create and Manage Resources with Templates
- CloudTrail — Track User Activity and API Usage
- Config — Track Resource Inventory and Changes
- OpsWorks — Automate Operations with Chef
- Service Catalog — Create and Use Standardized Products
- Trusted Advisor — Optimize Performance and Security

Security & Identity
- Identity & Access Management — Manage User Access and Encryption Keys
- Directory Service — Host and Manage Active Directory
- Inspector PREVIEW — Analyze Application Security
- WAF — Filter Malicious Web Traffic

Internet of Things
- AWS IoT BETA — Connect Devices to the cloud

Mobile Services
- Mobile Hub BETA — Build, Test, and Monitor Mobile apps
- Cognito — User Identity and App Data Synchronization
- Device Farm — Test Android, Fire OS, and iOS apps on real devices in the Cloud
- Mobile Analytics — Collect, View and Export App Analytics
- SNS — Push Notification Service

Application Services
- API Gateway — Build, Deploy and Manage APIs
- AppStream — Low Latency Application Streaming
- CloudSearch — Managed Search Service
- Elastic Transcoder — Easy-to-use Scalable Media Transcoding
- SES — Email Sending Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application Components

Enterprise Applications

EC2 / AWS main functions

- EC2 (Elastic Compute Cloud)
  - Instances
    - Servers of various sizes
  - AMIs (Amazon Machine Images)
    - Server images
  - Elastic Block Storage (EBS)
    - Virtualized Hard drives
  - VPC (Virtual Private Cloud)
    - Secure network space
- S3 (Simple Storage Solution)
  - "Buckets" of data
  - Longer term storage of data

Flintrock

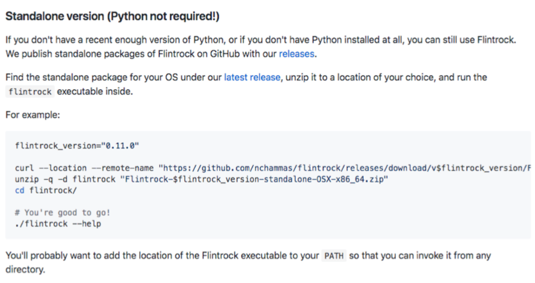
flintrock

license Apache 2.0 | build passing | chat on gitter

Watch @nchammas's talk on Flintrock at Spark Summit East 2016: talk / slides

Flintrock is a command-line tool for launching Apache Spark clusters.

Installing standalone version



Standalone version (Python not required!)

If you don't have a recent enough version of Python, or if you don't have Python installed at all, you can still use Flintrock. We publish standalone packages of Flintrock on GitHub with our releases.

Find the standalone package for your OS under our latest release, unzip it to a location of your choice, and run the `flintrock` executable inside.

For example:

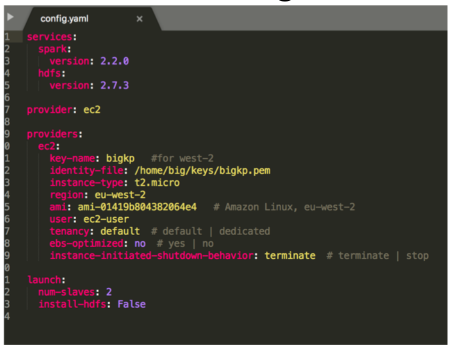
```
flintrock_version="0.11.0"

curl --location --remote-name "https://github.com/nchammas/flintrock/releases/download/v$flintrock_version/F
unzip -q -d flintrock "Flintrock-$flintrock_version-standalone-OSX-x86_64.zip"
cd flintrock/

# You're good to go!
./flintrock --help
```

You'll probably want to add the location of the Flintrock executable to your `PATH` so that you can invoke it from any directory.

Flintrock Launching a cluster

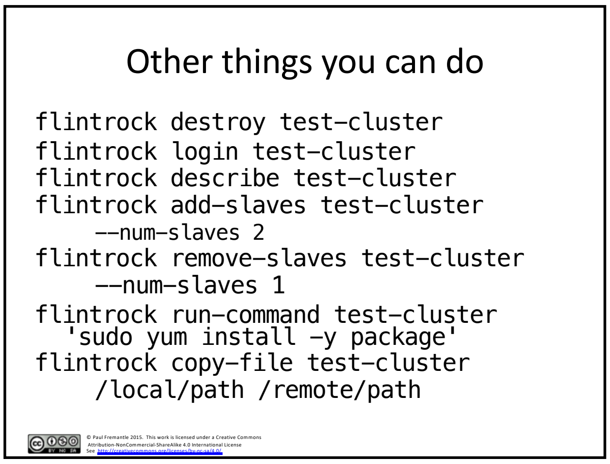
```
flintrock launch test-cluster \
    --num-slaves 1 \
    --spark-version 2.2.0 \
    --ec2-key-name key_name \
    --ec2-identity-file /path/to/key.pem \
    --ec2-ami ami-a4c7edb2 \
    --ec2-user ec2-user
```

Flintrock configure



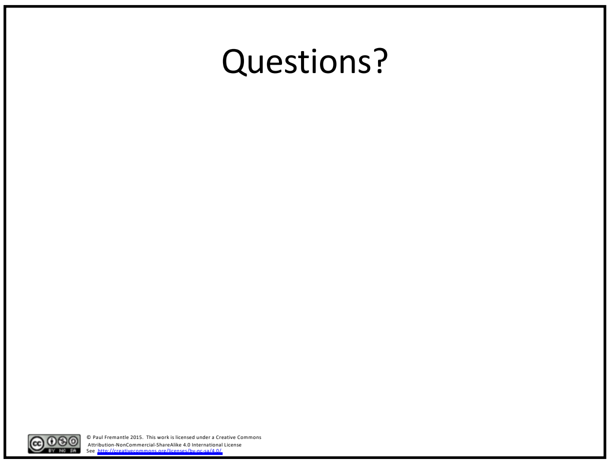
config.yaml

```
services:
  spark:
    version: 2.2.0
  hdfs:
    version: 2.7.3

provider: ec2

providers:
  ec2:
    key-name: bigkp   #for west-2
    identity-file: /home/big/keys/bigkp.pem
    instance-type: t2.micro
    region: eu-west-2
    ami: ami-01419b804382064e4   # Amazon Linux, eu-west-2
    user: ec2-user
    tenancy: default  # default | dedicated
    ebs-optimized: no  # yes | no
    instance-initiated-shutdown-behavior: terminate  # terminate | stop

launch:
  num-slaves: 2
  install-hdfs: False
```



Other things you can do

```
flintrock destroy test-cluster
flintrock login test-cluster
flintrock describe test-cluster
flintrock add-slaves test-cluster
    --num-slaves 2
flintrock remove-slaves test-cluster
    --num-slaves 1
flintrock run-command test-cluster
  'sudo yum install -y package'
flintrock copy-file test-cluster
    /local/path /remote/path
```



Questions?