Hadoop & Spark – Using Amazon EMR

Michael Hanisch, AWS Solutions Architecture
Agenda

Why did we build Amazon EMR?

What is Amazon EMR?

How do I get started using Amazon EMR?

Q&A
Why did we build Amazon EMR?
Hadoop: The Framework for Big Data Processing

Can process huge amounts of data

Scalable and fault tolerant

Respects multiple data formats

Supports an ecosystem of tools with a robust community

Executes batch and real-time analytics
Common Customer Challenges

Must purchase, provision, deploy and manage the infrastructure for Hadoop

Deploying, configuring and administering the Hadoop clusters can be difficult, expensive and time-consuming
What is Amazon Elastic MapReduce?
Amazon EMR Benefits

Simplifies Big Data Processing on a Managed Hadoop Framework

**Low Cost**
Pay an hourly rate for every instance hour you use

**Elastic**
Easily add or remove capacity

**Easy to use**
Launch a cluster in minutes

**Reliable**
Spend less time tuning and monitoring your cluster

**Secure**
Managed firewall settings

**Flexible**
You control every cluster
Amazon EMR is Elastic

Dynamically change number of compute resources

Provision one or thousands of instances to Process Data at any scale

Use CloudWatch to alert for scaling needs
Easy to Use

Launch a cluster in minutes through console

EMR sets up your cluster, provisions nodes, and configures Hadoop so you can focus on your analysis
Low Cost

Low Hourly Pricing for On-demand Instance Hours

Name your Price with Amazon EC2 Spot Instance Integration

Use EC2 Reserved Instances to reserve capacity and further lower your costs
Reliable

- Monitors cluster health
- Retries failed tasks
- Automatically replaces under-performing instances
Secure

Automatically configures firewall settings to control network access

Managed security groups for instances or specify your own

Launch in the Amazon Virtual Private Cloud (VPC)

Support for server-side & client-side encryption
Flexible

Completely control your cluster including customizations

Use data sources from Amazon S3, HDFS, Redshift, RDS, Kinesis and DynamoDB

Supports Apache & MapR Hadoop distributions

Runs popular Hadoop tools and frameworks (Hive, Hbase, Pig, Mahout, Spark, Presto)
How do I get started using Amazon EMR?
Easy to start

AWS Management Console

Command Line

Or use the Amazon EMR API with your favorite SDK.
How to use Amazon EMR

1. Upload your application and data to S3

2. Configure your cluster: Choose Hadoop distribution, number and type of nodes, applications (Hive/Pig/Hbase)

3. Launch your cluster using the console, CLI, SDK, or APIs

4. Retrieve your output results from S3
Configuration & Customization
Choose your instance types

Try different configurations to find your optimal architecture.

General
- m1 family
- m3 family

CPU
- c3 family
- cc1.4xlarge
- cc2.8xlarge

Memory
- m2 family
- r3 family

Disk/IO
- d2 family
- i2 family

Batch process
Machine learning
Spark and interactive
Large HDFS
Resizable clusters

Easy to add and remove compute capacity on your cluster.

Match compute demands with cluster sizing.
Core Nodes

- Run TaskTrackers (Compute)
- Run DataNode (HDFS)
Core Nodes

- Can add core nodes
- More HDFS space
- More CPU/mem
Amazon EMR Task Nodes

Run TaskTrackers

No HDFS

Reads from core node

HDFS

Amazon EMR cluster

Master instance group

Core instance group

Task instance group
Amazon EMR Task Nodes

Can Add Task Nodes
Amazon EMR Task Nodes

More CPU power
More memory
More network throughput
Amazon EMR Task Nodes

You can remove Task Nodes
Amazon EMR Task Nodes

Save Cost when Processing power is not required
Easy to use Spot Instances

Meet SLA at predictable cost

On-demand for core nodes
Standard Amazon EC2 pricing for on-demand capacity

Exceed SLA at lower cost

Spot Instances for task nodes
Up to 86% lower on average off on-demand pricing
Serial Processing

COST: 4h x $2.1 = $8.4
RENDERING TIME: 4h
1 Machine for 10 Hours ≈ 10 Machines for 1 Hour
Parallel Processing

COST: 2 x 2h x $2.1 = $8.4

RENDERING TIME: 2h
Embarrassingly Parallel Processing

COST: 4 x 1h x $2.1 = $8.4
RENDERING TIME: 1h
Pick the right version
Pick your Hadoop Version & Applications

EMR < 4.x:
- AMI version had fixed Hadoop version, Hive version etc.
- Additional applications via **Bootstrap Actions**
  Lots of different ones available from EMR & on github
- Configuration via Bootstrap Actions

https://github.com/awslabs/emr-bootstrap-actions
Pick your Hadoop Version & Applications

Starting with EMR 4.x:
- Release tags are independent of AMIs
- Pick from supported applications directly
  Based on Apache BigTop
- Configuration objects
  unified JSON syntax, mapped to configuration files

Recommendation: Use EMR 4.x by default!
Transient or persistent cluster?
Transient or persistent cluster?

Storing data on Amazon S3 give you more flexibility to run & shutdown Hadoop clusters when you need to.
Amazon S3 as your persistent data store

Amazon S3
- Designed for 99.999999999% durability
- Separate compute and storage

Resize and shut down Amazon EMR clusters with no data loss

Point multiple Amazon EMR clusters at same data in Amazon S3
Amazon EMR example #1: Batch processing

GBs of logs pushed to Amazon S3 hourly

Input and output stored in Amazon S3

Daily Amazon EMR cluster using Hive to process data

250 Amazon EMR jobs per day, processing 30 TB of data

http://aws.amazon.com/solutions/case-studies/yelp/
Amazon EMR example #2: Long-running cluster

Data pushed to Amazon S3

24/7 Amazon EMR cluster running HBase holds last 2 years' worth of data

Daily Amazon EMR cluster Extract, Transform, and Load (ETL) data into database

Front-end service uses HBase cluster to power dashboard with high concurrency
Amazon EMR example #3: Interactive query

TBs of logs sent daily → Logs stored in Amazon S3 → Amazon EMR cluster using Presto for ad hoc analysis of entire log set

Interactive query using Presto on multipetabyte warehouse

Steps & Clusters
Applications
MR job

MapReduce

HDFS

Amazon S3

Local filesystem
Local filesystem

HDFS

Amazon S3

Local filesystem

MR job

MapReduce v2: YARN

HDFS

Spark SQL
Amazon S3 and HDFS
Amazon S3 has near linear scalability

S3 Streaming Performance
- 100 VMs; 9.6GB/s; $26/hr
- 350 VMs; 28.7GB/s; $90/hr

34 secs per terabyte

Reader Connections

GB/Second
EMRFS makes it easier to leverage Amazon S3

Better performance and error handling options

Transparent to applications – just read/write to “s3://”

Consistent view
  • For consistent list and read-after-write for new puts

Support for Amazon S3 server-side and client-side encryption

Faster listing using EMRFS metadata
EMRFS support for Amazon S3 client-side encryption

Amazon S3

(client-side encrypted objects)

Amazon S3

EMRFS enabled for Amazon S3 client-side encryption

Amazon S3 encryption clients

Key vendor (AWS KMS or your custom key vendor)
Fast listing of Amazon S3 objects using EMRFS metadata

- List and read-after-write consistency
- Faster list operations

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Without Consistent Views</th>
<th>With Consistent Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000,000</td>
<td>147.72</td>
<td>29.70</td>
</tr>
<tr>
<td>100,000</td>
<td>12.70</td>
<td>3.69</td>
</tr>
</tbody>
</table>

*Tests using a single node cluster with a m3.xlarge instance.
Optimize to leverage HDFS

Iterative workloads
  • If you’re processing the same dataset more than once

Disk I/O intensive workloads

Persist data on Amazon S3 and use S3DistCp to copy to HDFS for processing.
Running Spark on Amazon EMR
Launch the latest Spark version

July 15 – Spark 1.4.1 GA release
July 24 – Spark 1.4.1 available on Amazon EMR

September 9 – Spark 1.5.0 GA release
September 30 – Spark 1.5.0 available on Amazon EMR

< 3 week cadence with latest open source release
Amazon EMR runs Spark on YARN

- Dynamically share and centrally configure the same pool of cluster resources across engines
- Schedulers for categorizing, isolating, and prioritizing workloads
- Choose the number of executors to use, or allow YARN to choose (dynamic allocation)
- Kerberos authentication

### Applications
- Pig, Hive, Cascading, Spark Streaming, Spark SQL

### YARN
- Cluster Resource Management

### Storage
- S3, HDFS

### Batch
- MapReduce

### In Memory
- Spark
Create a fully configured cluster in minutes

AWS Management Console

AWS Command Line Interface (CLI)

Or use an AWS SDK directly with the Amazon EMR API
Or easily change your settings

Edit software settings (optional)


Change settings  Enter configuration  Load JSON from S3

Many storage layers to choose from

- Amazon DynamoDB
- Amazon RDS
- Elasticsearch
- Amazon S3
- Amazon EMR
- Amazon Redshift
- Amazon Redshift Copy From HDFS
- Amazon Kinesis
- Spark JDBC Data Source w/ Spark SQL
- EMR-DynamoDB connector
- EMR File System (EMRFS)
- Streaming data connectors
- Elasticsearch connector
Decouple compute and storage by using S3 as your data layer

S3 is designed for 11 9’s of durability and is massively scalable
Easy to run your Spark workloads

Submit a Spark application

Amazon EMR Step API

SSH to master node (Spark Shell)

Amazon EMR
Spark on YARN

YARN is Hadoop 2’s Cluster Manager
Spark Standalone Cluster Manager is FIFO, 100% Core Allocation
Not ideal for multi-tenancy
YARN Default is ‘Capacity Scheduler’
Focus on Throughput
Configure Queues
‘Fair Scheduler’ also available
Focus on even distribution
Why run Spark on YARN?

- YARN allows you to dynamically share and centrally configure the same pool of cluster resources between all frameworks that run on YARN. You can throw your entire cluster at a MapReduce job, then use some of it on an Impala query and the rest on Spark application, without any changes in configuration.

- You can take advantage of all the features of YARN schedulers for categorizing, isolating, and prioritizing workloads.

- Spark standalone mode requires each application to run an executor on every node in the cluster, whereas with YARN you choose the number of executors to use.

- YARN is the only cluster manager for Spark that supports security. With YARN, Spark can run against Kerberized Hadoop clusters and uses secure authentication between its processes.
Task Scheduler

- Supports general task graphs
- Pipelines functions where possible
- Cache-aware data reuse & locality
- Partitioning-aware to avoid shuffles

Stage 1
- A: join
- B: groupBy

Stage 2
- C: map
- D: filter
- E: join

Stage 3
- F: = cached partition

= RDD
Secure Spark clusters – encryption at rest

On-Cluster
HDFS transparent encryption (AES 256)
[new on release emr-4.1.0]
Local disk encryption for temporary files using LUKS encryption via bootstrap action

Amazon S3
EMRFS support for Amazon S3 client-side and server-side encryption (AES 256)
Secure Spark clusters – encryption in flight

Internode communication on-cluster
Blocks are encrypted in-transit in HDFS when using transparent encryption

Spark’s Broadcast and FileServer services can use SSL. BlockTransferService (for shuffle) can’t use SSL (SPARK-5682).

S3 to Amazon EMR cluster
Secure communication with SSL

Objects encrypted over the wire if using client-side encryption
Secure Spark clusters – additional features

Permissions:

- **Cluster level**: IAM roles for the Amazon EMR service and the cluster
- **Application level**: Kerberos (Spark on YARN only)
- **Amazon EMR service level**: IAM users

**Access**: VPC, security groups

**Auditing**: AWS CloudTrail
Thank you!