Hadoop modules

عادل قاضی خانی
هیئت علمی گروه کامپیوتر دانشگاه امام رضا (ع)

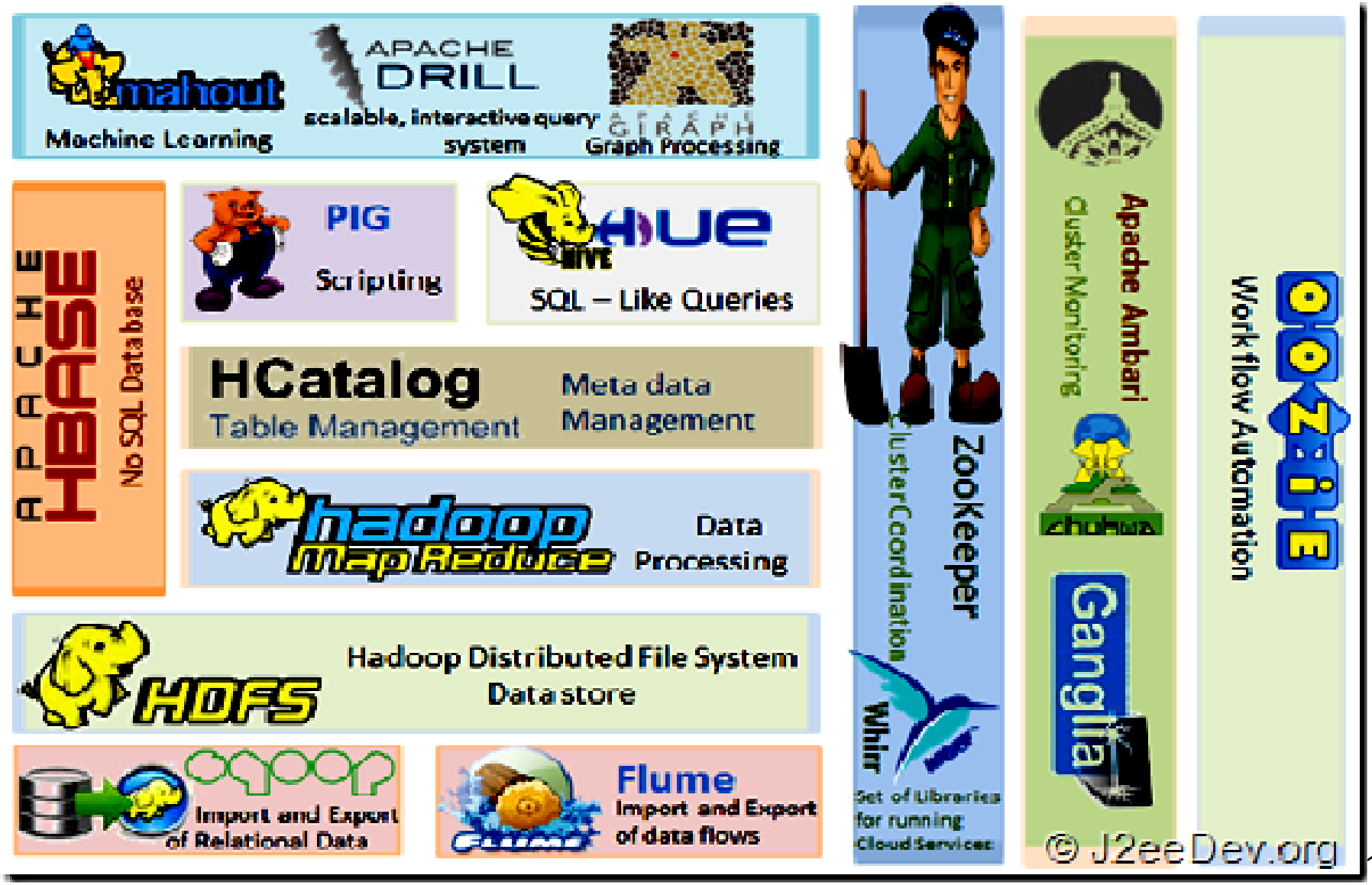
تعریف

- هدوپ بستری برای **ذخیره سازی داده های حجیم** بر روی **خوشه ای** [1] از کامپیوترهای ساده و **انجام عملیات روی داده ها** است

- هدوپ **اجزای** متنوعی دارد

1-cluster

اجزای هدوپ


Oozie
Work flow Automation
Apache Ambari
Cluster Monitoring
Ganglia
© J2eeDev.org
Zookeeper
Cluster Coordination
Whirr
Set of Libraries for running Cloud Services
mahout
Machine Learning
APACHE DRILL
scalable, interactive query system
APACHE GIRAPH
Graph Processing
APACHE HBASE
No SQL Data base
PIG
Scripting
Hue
SQL – Like Queries
HCatalog
Table Management
Meta data Management
hadoop MapReduce
Data Processing
HDFS
Hadoop Distributed File System
Data store
Sqoop
Import and Export of Relational Data
Flume
Import and Export of data flows

HDFS

بستر ذخیره سازی

ساختار HDFS

- معماری **Master/slave** دارد
- سرویس Master بنام **NameNode** دسترسی به داده ها را کنترل می کند
- سرویس Slave بنام **DataNode** در هر گره خوشه وجود دارند و عملیات خواندن و نوشتن داده ها را مدیریت می کنند

1-Map
2-Reduce

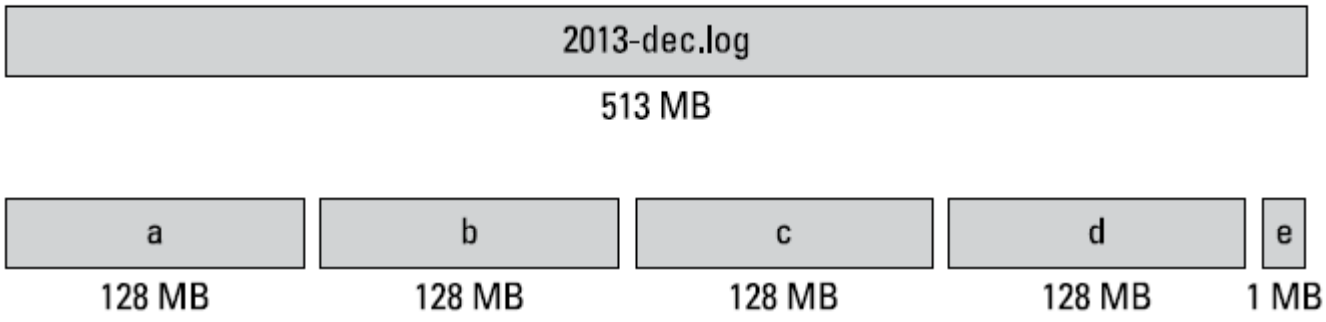
مقدمه

- در HDFS امکان مقیاس پذیری ذخیره سازی میان هزاران گره slave وجود دارد
- میانگین اندازه فایلها می تواند 500MB باشد
- مدل دسترسی به داده: یکبار بنویس، غالباً بخوان (Write Once, Often Read)

1-Hadoop Distributed File System

بلوکهای داده

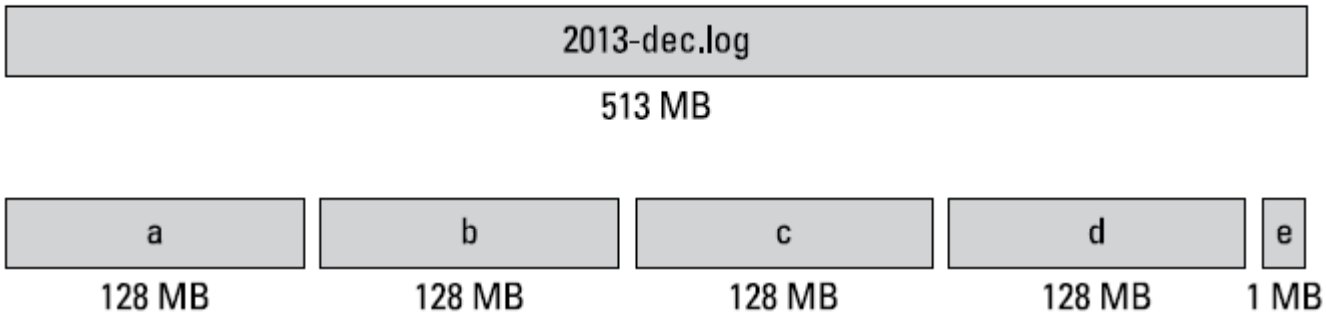
- وقتی ذخیره سازی در HDFS انجام می شود سیستم آنرا به تعدادی بلوک تقسیم و در نودهای slave ذخیره می کند



1-Hadoop Distributed File System

بلوکهای داده

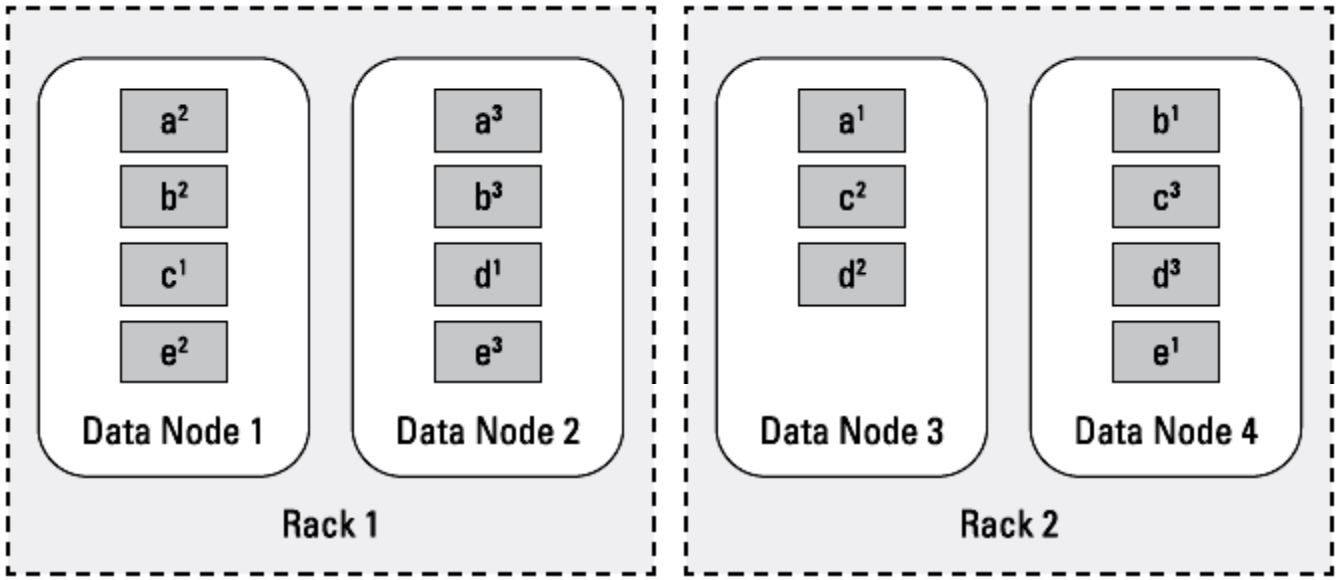
- اندازه بلوکها قابل تنظیم است
- معمولاً 128MB



1-Hadoop Distributed File System

تکرار بلوکهای داده

- بدلیل ساده بودن سخت افزار ذخیره سازی و امکان خرابی، از هر بلوک داده سه نسخه ذخیره می شود



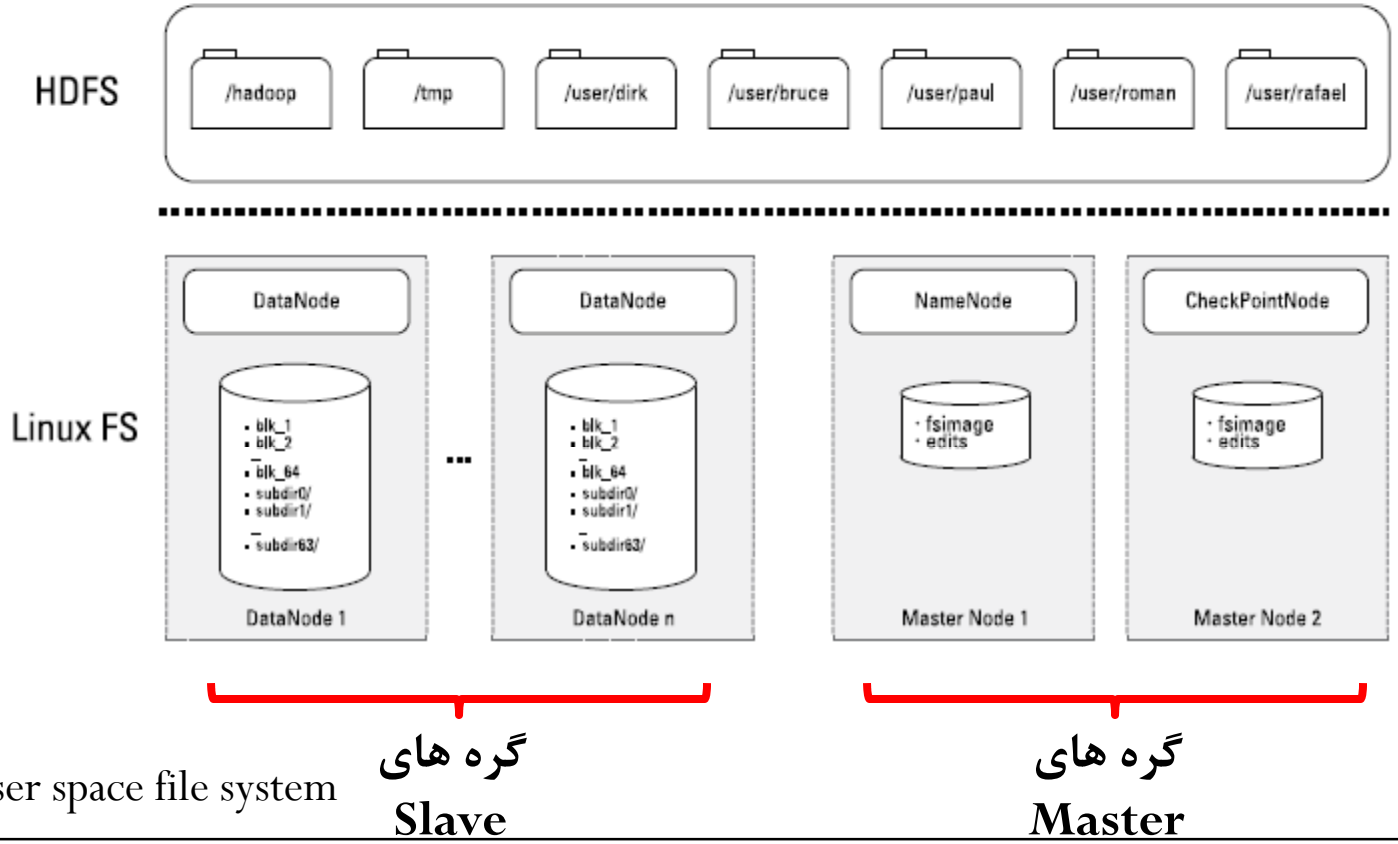
Replicating

خراب شدن دیسک

- نظارت آن بر عهده Name Node ها است
- اگر دیسکی از بین برود Name Node آنرا تشخیص می دهد و نسخه های بلوک کم شده را تولید می کند و در زمان برگشت دیسک نسخه های بلوک اضافی را حذف می کند

HDFS

- یک سیستم فایل در فضای کاربر



گره های Slave

گره های Master

User space file system

گره های slave

- Datanode
- داده ها در آنها قرار دارد
- تکرار داده ها بدلیل استفاده از سخت افزارهای معمولی
- سیستم فایل Ext3

گره های Master

- Namenode
- مدیریت بلوکهای داده
- در زمان بارگزاری fsimage و editlog را بارگزاری می شوند. همچنین یک سرویس بصورت دائمی از Datanode ها گزارش می گیرد

گره های Master

- fsimage: فایلی که در آن تمام اطلاعات مربوط به اینکه هر فایلی در کدام بلوکها ذخیره شده است به همراه نسخه ها و همچنین ویژگی های سیستم فایل در آن قرار دارد

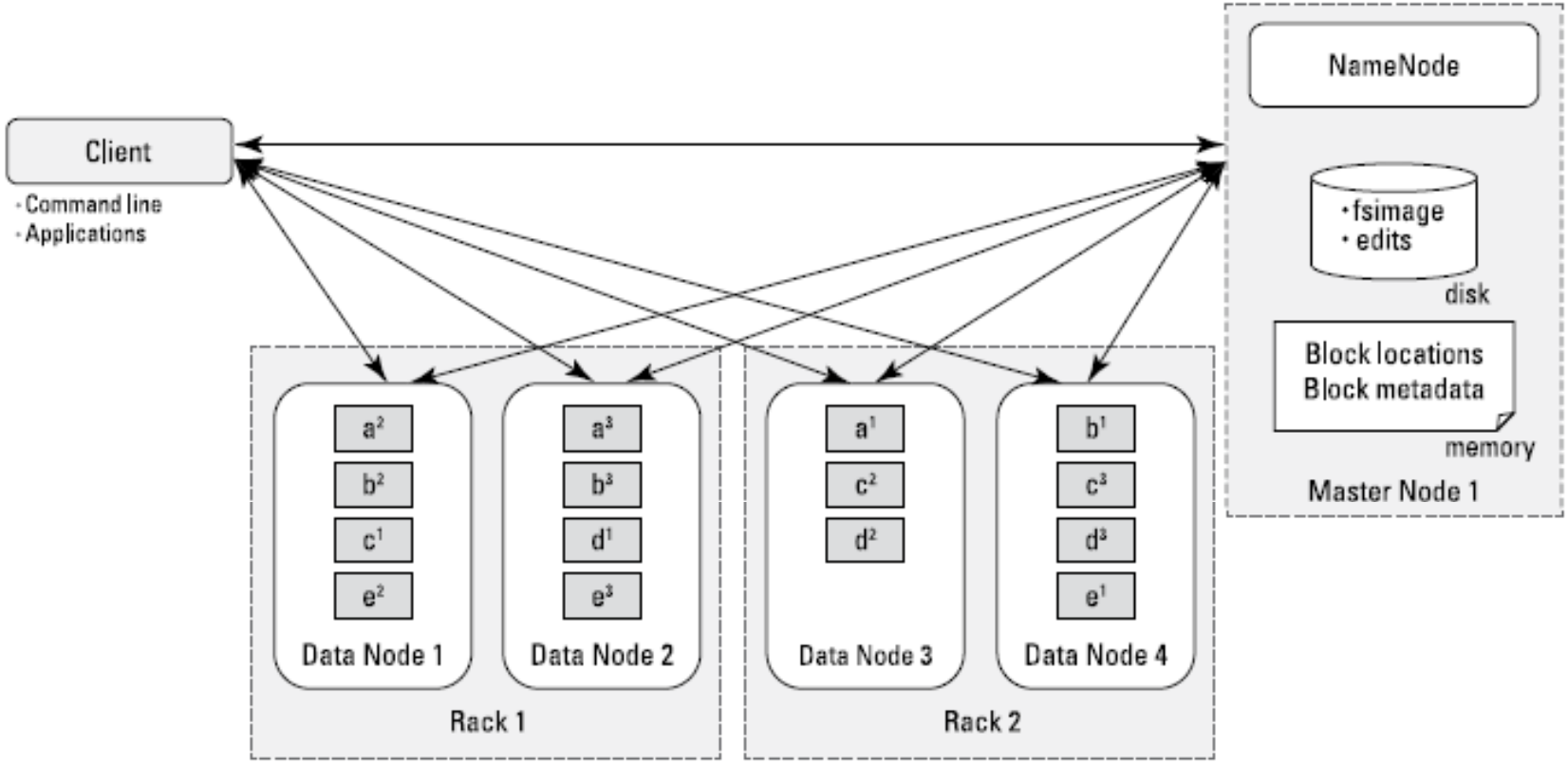
- edits : فایلی که تمام عملیات روی فایلها در آن ذخیره می شود

گره های Master

- هر سه ثانیه یکبار یک سرویس در ِDatanode ها پیامی مبنی بر زنده بودن برای Namenode ارسال می کنند

- هر ۶ ساعت یکبار یک سرویس در ِDatanode ها پیامی که بیانگر بلوکهای موجود در آن است برای Namenode ارسال می کنند

ساختار کلی


NameNode
Client
· Command line
· Applications
· fsimage
· edits
disk
Block locations
Block metadata
memory
Master Node 1
a2
b2
c1
e2
Data Node 1
a3
b3
d1
e3
Data Node 2
a1
c2
d2
Data Node 3
b1
c3
d3
e1
Data Node 4
Rack 1
Rack 2

فرایند نوشتن در HDFS

- Client یک درخواست ایجاد فایل جدید ارسال می کند
- Namenode مجوز برای دسترسی به خوشه را ارسال می کند
- Client عملیات نوشتن را بر روی slave ها انجام می دهد
- Namenode روی عملیات نظارت دارد.
- یک سرویس کپی ها را در مکانهای دیگر می نویسد
- بعد از اتمام نوشتن کپی ها توسط سرویسها فایل توسط Client بسته می شود

فرایند خواندن در HDFS

- Client یک درخواست خواندن ارسال می کند
- Namenode بلوکها را مشخص می کند
- Client عملیات خواندن با توجه به آدرسهای تعیین شده توسط Namenode انجام می دهد

اعمال تغییرات edits

- همانطور که بیان شد هر گونه تغییرات در فایلها در edits ثبت می شود. اما بعد از مدتی تغییرات باید در fsimage اعمال شود (checkpoint)

- این کار را خود NameNode انجام نمی دهد بخاطر امکان خرابی

اعمال تغییرات editlog

- عمل checkpoint توسط یکی از نودهای زیر انجام می شود
- Checkpoint node
- Backup node
- Standby Namenode

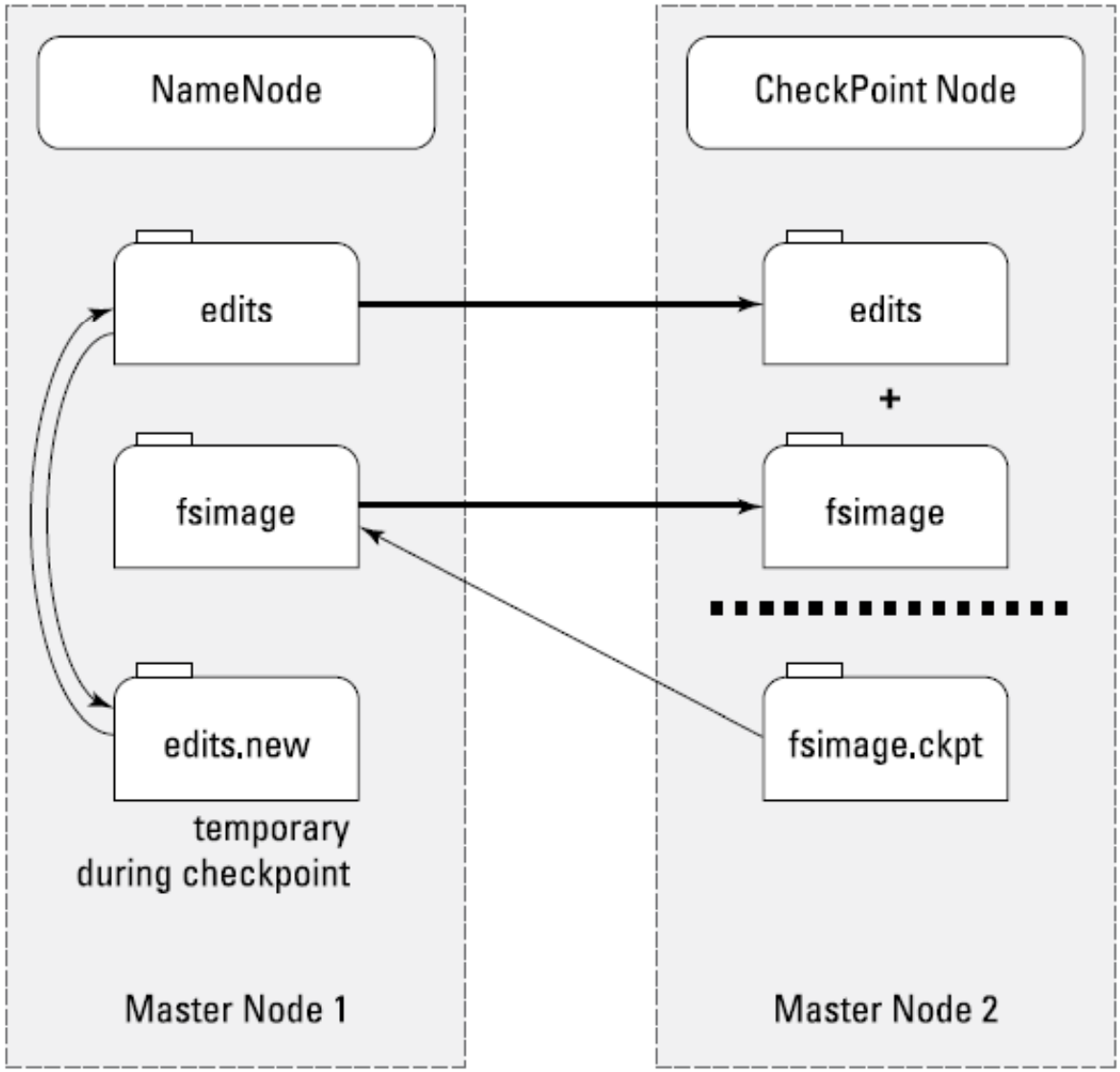
فرایند checkpoint

- در زمان checkpoint ، Namenode یک فایل جدید بنام edits.new برای تغییرات جدید درست می کند
- فایل edits قبلی تغییر دیگری را نمی پذیرد و به همراه fsimage به سرویس checkpoint کپی می شود
- سرویس checkpoint دو فایل را با هم ترکیب می کند در یک فایل بنام fsimage.ckpt قرار می دهد

فرایند checkpoint ادامه

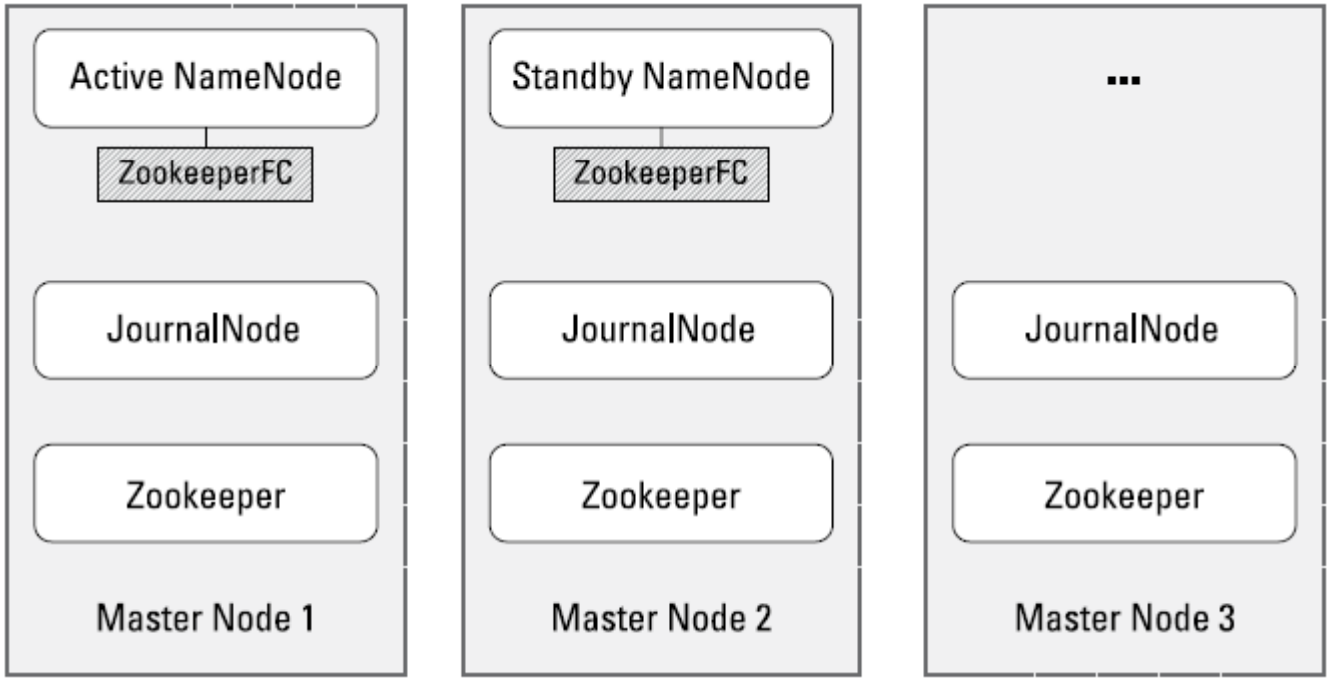
- سرویس checkpoint فایل fsimage.ckpt را در Namenode کپی می کند
- Namenode فایل fsimage را با fsimage.ckpt جایگرین می کند
- Namenode فایل edits.new را به edits تغییر نام می دهد

فرایند checkpoint



دسترس پذیری بالا

- برای افزایش دسترس پذیری Namenode می توان از Active و Standby ، Namenode استفاده کرد



Mapreduce

پردازش نگاشت-کاهش

MapReduce

- هر کار در MapReduce شامل دو مرحله **نگاشت**[1] و **کاهش**[2] است

- داده ها به شکل **کلید – مقدار**[3] هستند

1-Map
2-Reduce
3-key-value

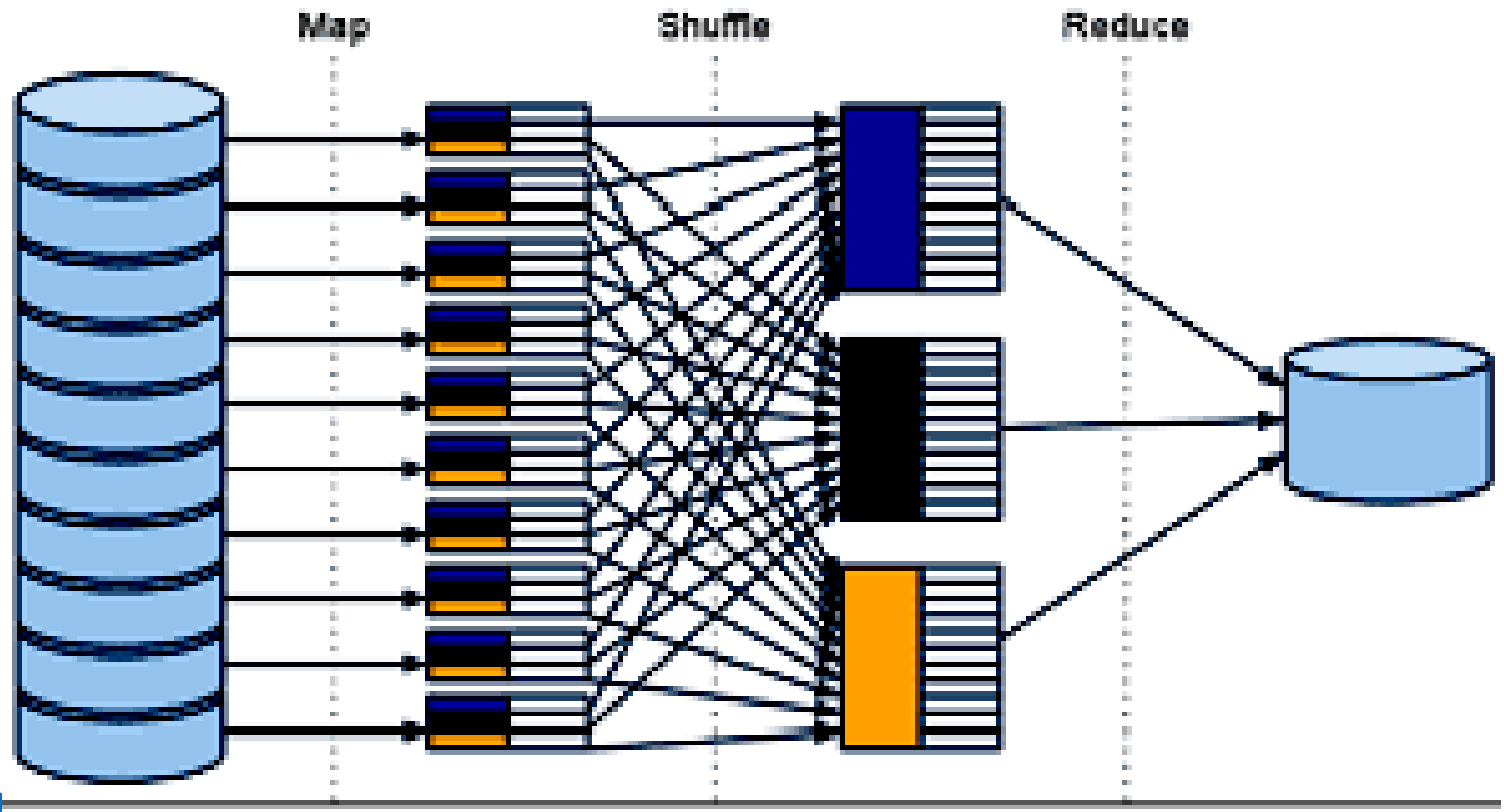
MapReduce

- در **مرحله نگاشت** داده ها به **بخشهای زیادی تقسیم می شود** و هر کدام به یک عمل نگاشت نسبت داده می شوند
- این اعمال نگاشت در **خوشه توزیع می شوند**
- هر عمل نگاشت داده های کلید-مقدار خود را پردازش می کند و یک داده میانی با فرمت کلید-مقدار تولید می کند

MapReduce

- تمام داده های میانی بر اساس **کلید مرتب شده** و به تعدادی عمل کاهش نسبت داده می شوند
- در **مرحله کاهش** هر عمل کاهش داده های خود را پردازش می کند و یک خروجی کلید-مقدار بوجود می آورد
- اعمال کاهش نیز در خوشه توزیع می شوند و خروجی را در HDFS می نویسند

MapReduce



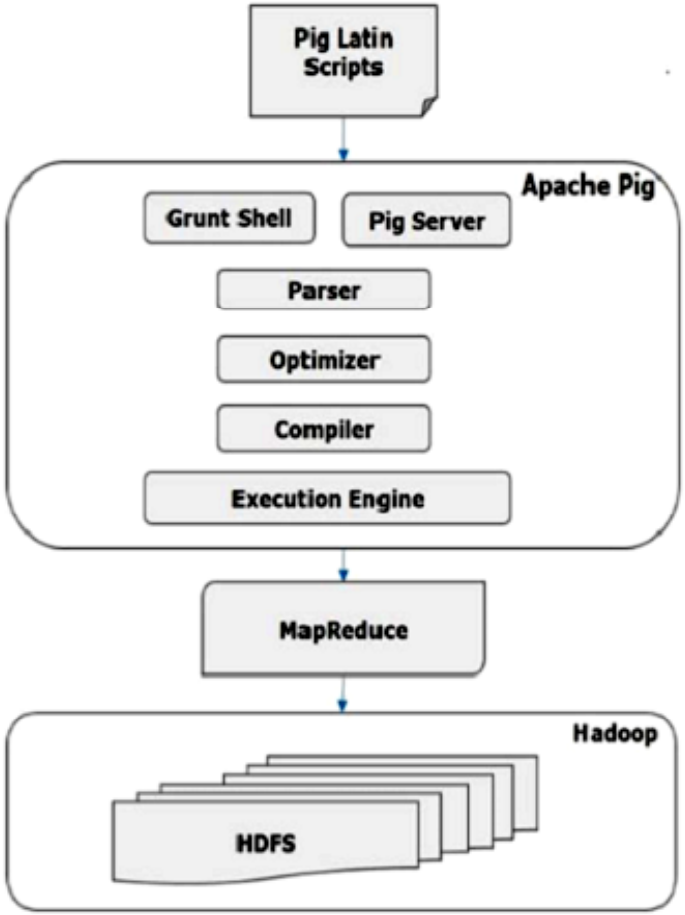
ماژول Mahout

- کتابخانه ای از ابزارهای **یادگیری ماشین**
- تشکیل شده از **سه بخش اصلی**:
- **طبقه‌بندی**
- **خوشه‌بندی**
- **سیستم‌های پیشنهاد دهنده**

ماژول Pig

- یک **زبان سطح بالا** برای نوشتن برنامه های کوتاه به منظور پردازش داده‌ها. دستورات این زبان بصورت نگاشت - کاهش اجرا می شوند.

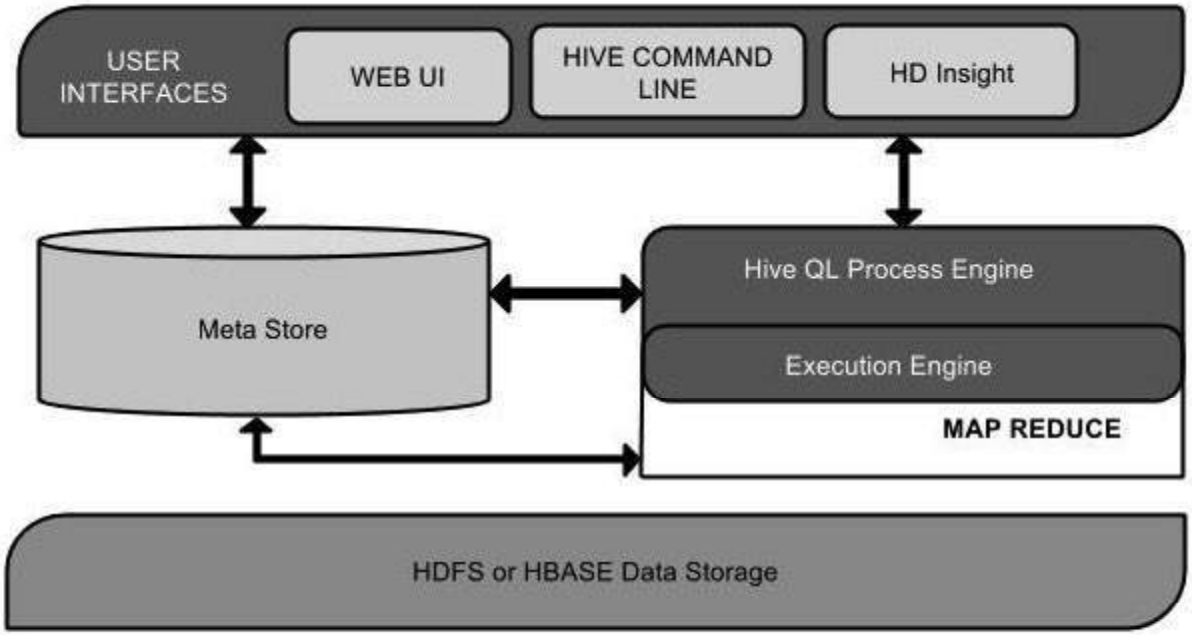
- معماری کلی



ماژول Hive

- ابزاری برای **نوشتن دستورات شبه SQL** در هدوپ. دستورات Hive بصورت نگاشت - کاهش اجرا می شوند.

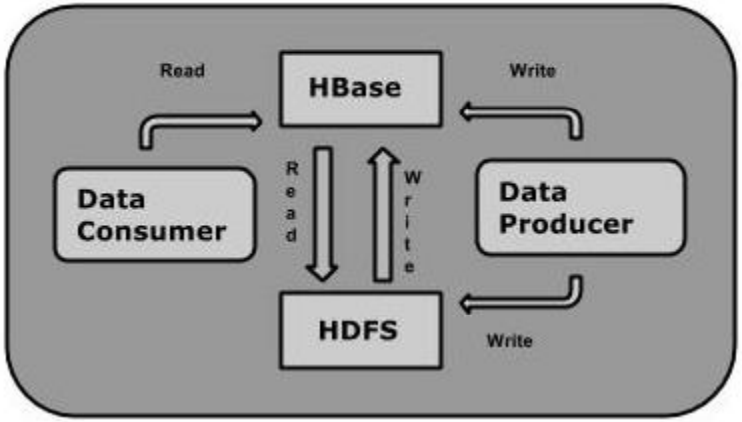
- معماری کلی



HBase

- **پایگاه داده ای غیر رابطه ای** که در سطحی بالاتر از HDFS امکان دسترسی مستقیم و سریع به داده ها را فراهم می کند.

- معماری کلی



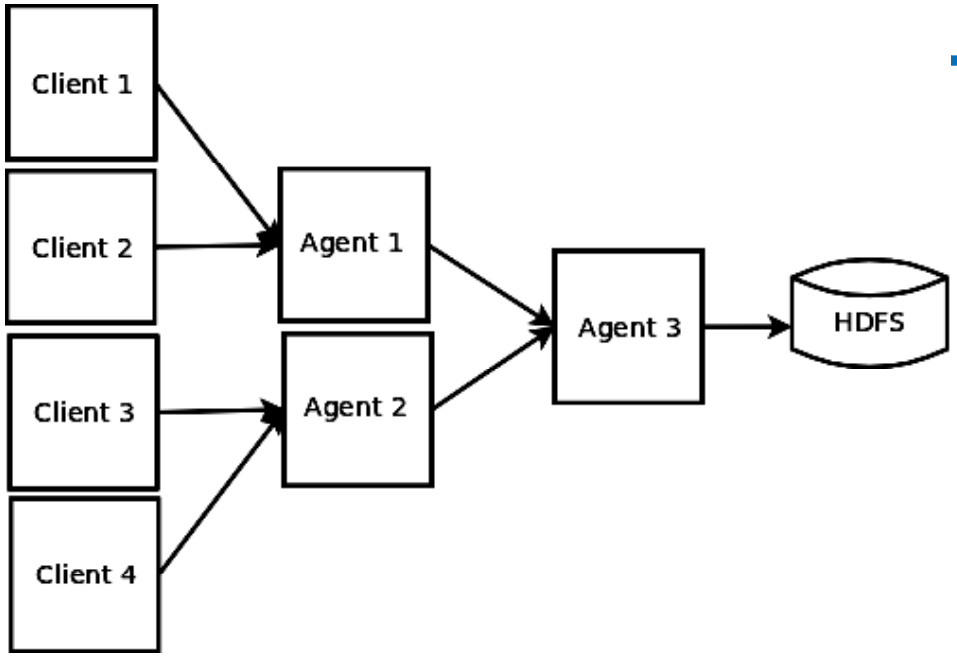
HCatalog

- ابزاری برای **مدیریت ذخیره سازی داده ها به شکل جدول** در هدوپ. داده های موجود در HDFS را بصورت جدولی ارائه می دهد. از انواع فرمت فایل پشتیبانی می کند.

- در Pig,Hive,Mapreduce می توان از آن استفاده کرد

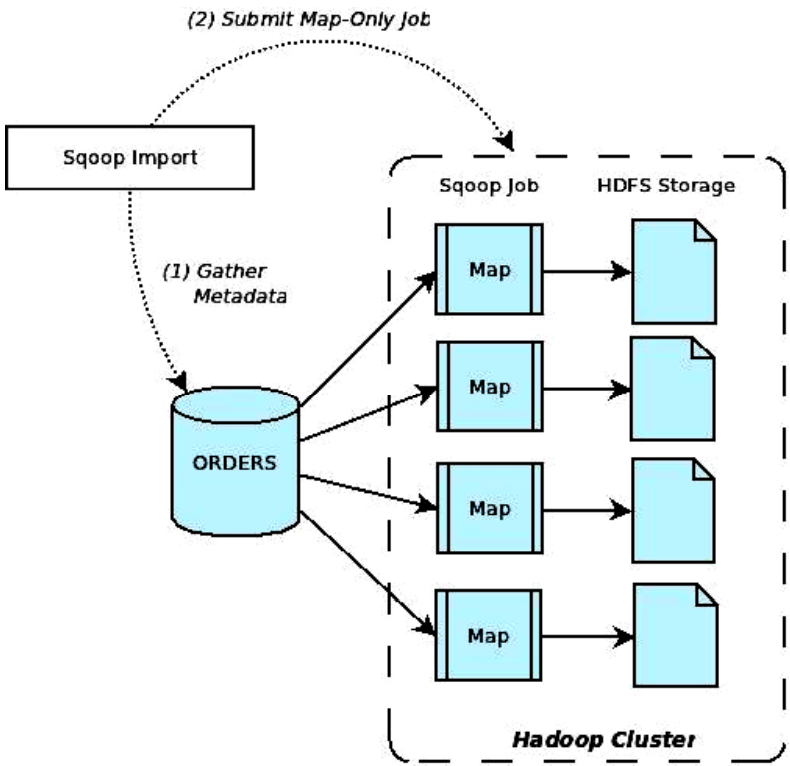
Flume

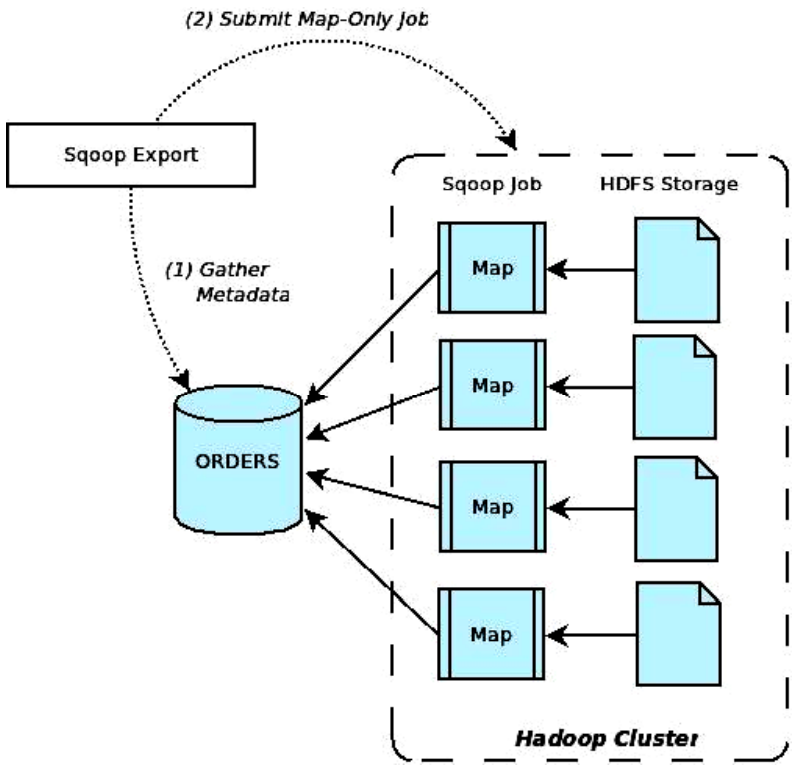
- سیستمی توزیع شده، قابل اعتماد و در دسترس **برای جمع آوری، تجمیع و انتقال داده های log از منابع مختلف به انبار داده مرکزی**



Sqoop

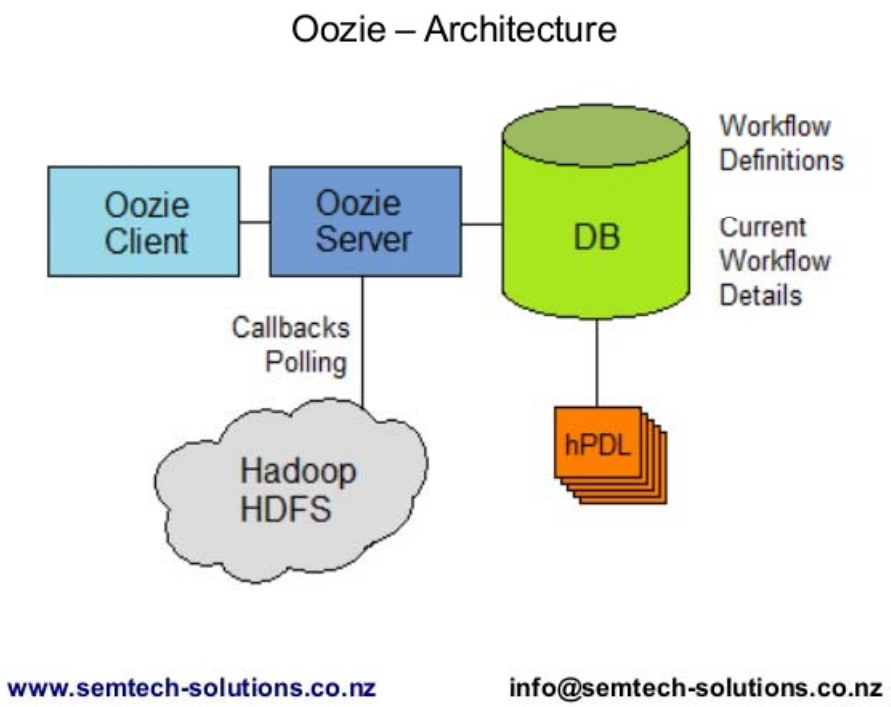
- ابزاری برای **ورود و خروج داده ها (import/export)** در هدوپ. وارد کردن داده ها از منابع مختلف خارجی و انتقال نتایج به نرم افزارهای دیگر





Oozie

- یک برنامه زمانبند برای مدیریت کارها در هدوپ



Ambari

- ابزاری برای **آماده‌سازی، مدیریت و کنترل خوشه** در هدوپ.

- **آماده‌سازی**: نصب و پیگربندی سرویس های هدوپ
- **مدیریت**: شروع، خاتمه و پیکربندی مجدد سرویسها
- **کنترل**: مشاهده وضعیت گره ها در خوشه

Zookeeper

- سرویسی برای **هماهنگی با کارایی بالا** در برنامه های توزیع شده.
- سرویسهایی از قبیل: نام‌گذاری، مدیریت پیکربندی، همزمانی،

خواندن و نوشتن داده ها

خواندن و نوشتن داده ها

- برای **تحلیل داده های حجیم** اولین مرحله **خواندن** آن و یا **نوشتن** (انجام تغییرات اولیه یا پیش پردازش) آنها

فشرده سازی داده ها

- بدلیل حجم بالای داده ها **یک ضرورت** است
- **Codec** ها مختلف (نرم افزارها)
- **فشرده سازی قابل تقسیم** که بستگی به codec دارد
- **Mapreduce** امکان کار با **فایلهای فشرده** را دارد
- **هزینه** فشرده سازی

فشرده سازی داده ها

- Codec هایی که هدوپ پشتیبانی می کند

Codec	File Extension	Splittable?	Degree of Compression	Compression Speed
Gzip	.gz	No	Medium	Medium
Bzip2	.bz2	Yes	High	Slow
Snappy	.snappy	No	Medium	Fast
LZO	.lzo	No, unless indexed	Medium	Fast

مدیریت فایلها

- **دستوراتی** برای دسترسی و عملیات روی فایلها در هدوپ وجود دارد
- هر فایل بصورت تعدادی بلوک در HDFS وجود دارد
- در ادامه چند دستور مهم ذکر می شود

مدیریت فایلها

- در دستورات برای **دسترسی به یک مسیر یا فایل** از URI استفاده می شود که می تواند مسیر کامل باشد یا کوتاه باشد

- مثلاً مسیر کامل
- *hdfs://namenodehost/parent/child*
- مسیر کوتاه اگر در فایل تنظیمات *hdfs://namenodehost* باشد
- */parent/child*

مدیریت فایلها

- **فرمت کلی** دستورات مدیریت فایل که شبیه Linux هستند
- *hadoop hdfs dfs* - دستور
- **مثلاً** ایجاد شاخه، کپی کردن، حذف
- *hadoop hdfs dfs -mkdir*
- *hadoop hdfs dfs -put*
- *hadoop hdfs dfs -rm*
- دستور برای تعیین بلوکهای فایلها
- hadoop fsck / -files –blocks

خواندن داده های جریانی

- **نرم افزار Flume** برای **تجمیع و انتقال داده های جریانی** از منابع مختلف به یک سرور مرکزی

- مثلاً: داده های log ها

نحوه کار Flume

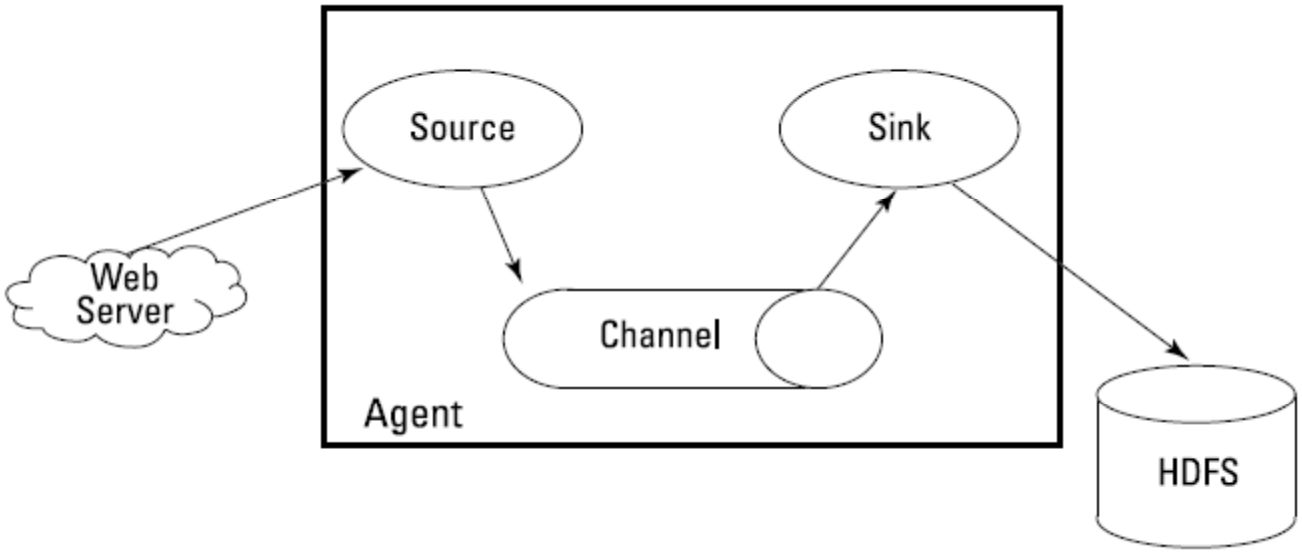
- بصورت **یک یا چند عامل** اجرا می شود

- هر عامل باید حداقل یکی از سه مؤلفه : **منبع؟، کانال؟ و شیر؟** را داشته باشد

source, channel, sink

بخشهای یک عامل

- **منبع**: داده ها را دریافت و به کانال می فرستند
- **کانال**: داده ها را بافر می کنند رابط بین منبع و شیر
- **شیر**: پردازش داده ها و تحویل به مقصد (مثل HDFS)



درباره عامل ها

- هر **عامل** می تواند **شامل چند** منبع، کانال و شیر باشد
- یک **منبع** می تواند به **چندین کانال بنویسد** ولی یک **شیر** فقط می تواند **از یک کانال** داده بگیرد
- هر عامل یک **ماشین مجازی جاوا** (JVM) است که **Flume** را اجرا می کند

درباره عامل ها – ادامه

- **بخش شیر** (**sink**) برای هر نود عامل، داده ها را برای **نودهای جمع کننده** که از چندین شیر ورودی می گیرند ارسال می کند تا بعد از تجمیع برای HDFS ارسال شوند

- **عاملها** می توانند به یکدیگر **زنجیر** شوند. یعنی مثلاً *sink* (شیر) یک عامل داده ها را برای *source* (منبع) یک عامل دیگر می فرستد

درباره عامل ها – ادامه

- تنظیمات مربوط به عاملها در یک فایل تنظیمات وارد می شود
- **مثال**

```
# Identify the components on agent shaman:
shaman.sources = netcat_s1
shaman.sinks = hdfs_w1
shaman.channels = in-mem_c1

# Configure the source:
shaman.sources.netcat_s1.type = netcat
shaman.sources.netcat_s1.bind = localhost
shaman.sources.netcat_s1.port = 44444

# Describe the sink:
shaman.sinks.hdfs_w1.type = hdfs
shaman.sinks.hdfs_w1.hdfs.path = hdfs://<path>
shaman.sinks.hdfs_w1.hdfs.writeFormat = Text
shaman.sinks.hdfs_w1.hdfs.fileType = DataStream
# Configure a channel that buffers events in memory:
shaman.channels.in-mem_c1.type = memory
shaman.channels.in-mem_c1.capacity = 20000
shaman.channels.in-mem_c1.transactionCapacity = 100

# Bind the source and sink to the channel:
shaman.sources.netcat_s1.channels = in-mem_c1
shaman.sinks.hdfs_w1.channels = in-mem_c1
```

درباره عامل ها – ادامه

- برای شروع به کار عامل می توان از دستور زیر استفاده کرد

```
flume-ng agent -f /<path to flume-agent.conf> -n shaman
```