

FIAP GRADUAÇÃO

Introdução ao Hadoop

Hadoop

HDFS e O Problema dos Discos



Node, Rack, Cluster

- Node
 - CPU + RAM + HardDisk – são os três elementos primários que formam um NODE.
- Rack
 - Coleção de Nodes
- Cluster
 - Racks interconectados pela rede.

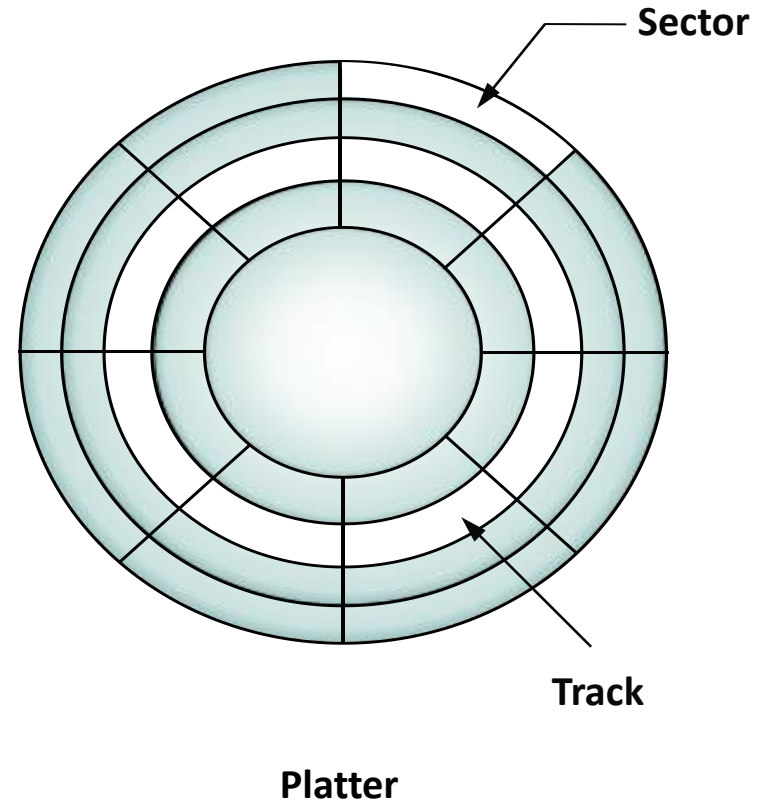
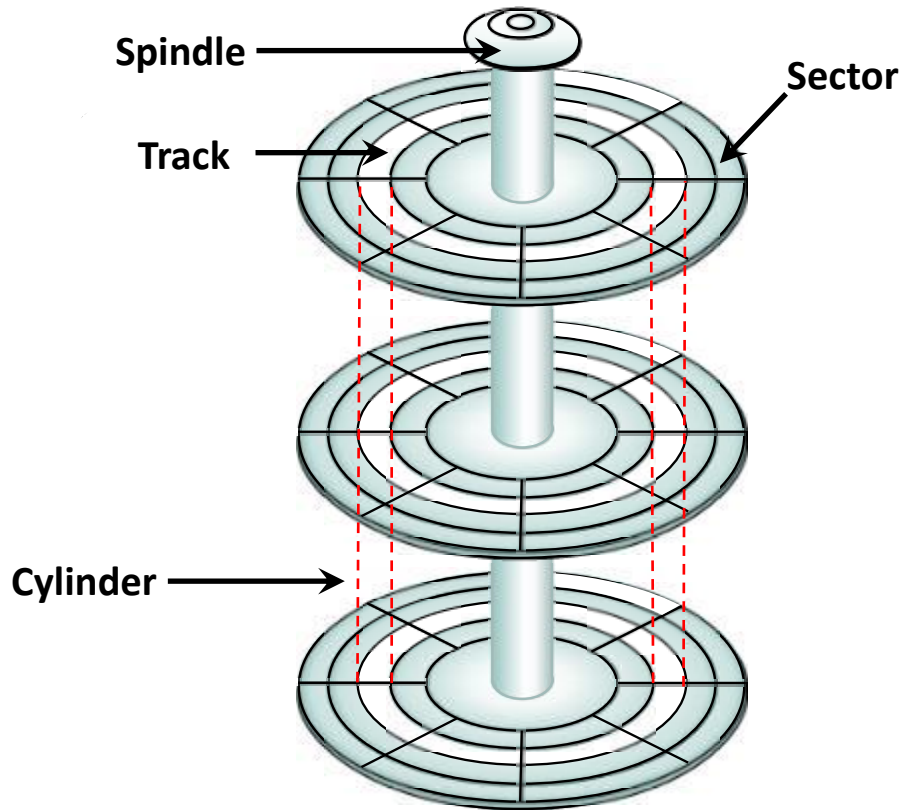


Node



Estrutura Física do Disco

FIAP



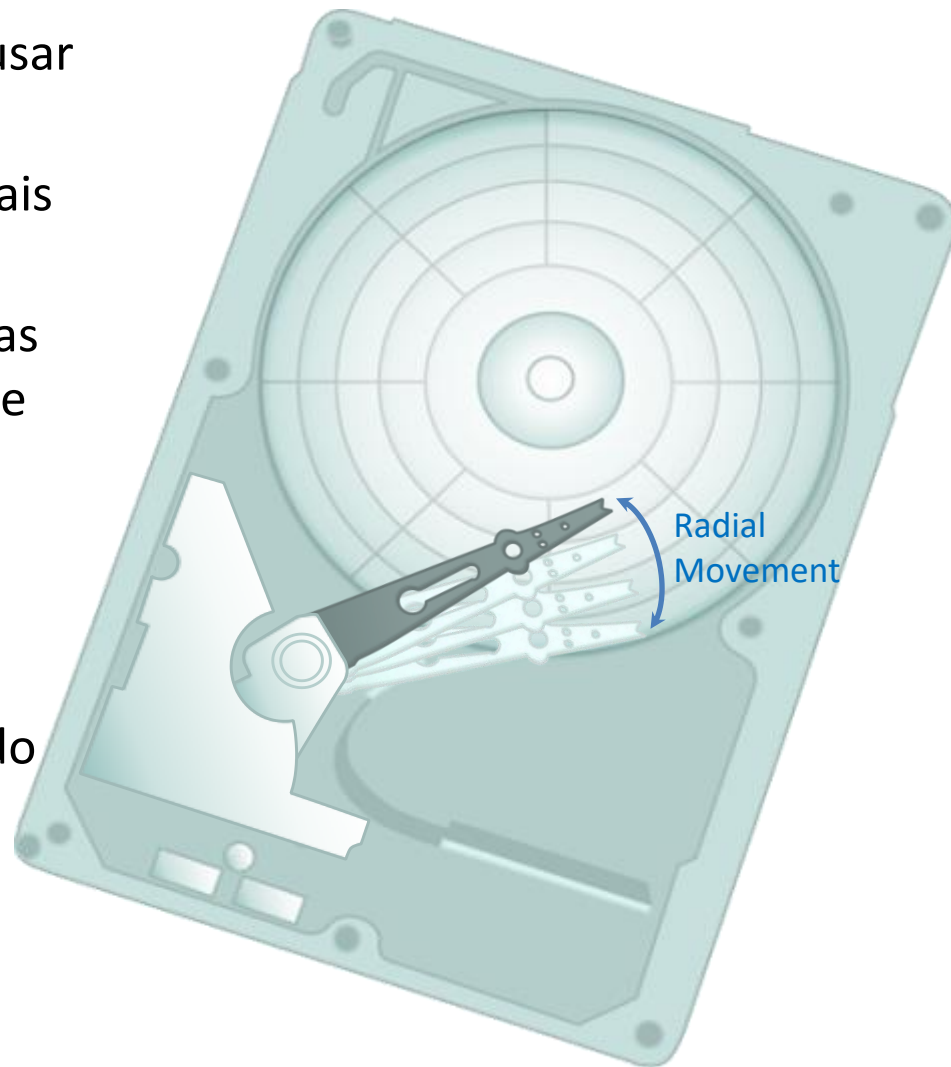
Desempenho do Drive do Disco FIAP

- Dispositivo eletromagnético
 - Afeta o desempenho geral do ambiente do sistema de armazenamento.
- Tempo de serviço do disco
 - Tempo gasto por um disco para completar uma solicitação de I/O:
 - Tempo de busca (Seek time)
 - Latencia rotacional (Rotational latency)
 - Taxa de transferencia de dados (Data transfer rate)

Desempenho do drive de disco= Tempo de busca +
Latencia rotacional +
Taxa de transferência de dados

Tempo de Busca (*Seek Time*)

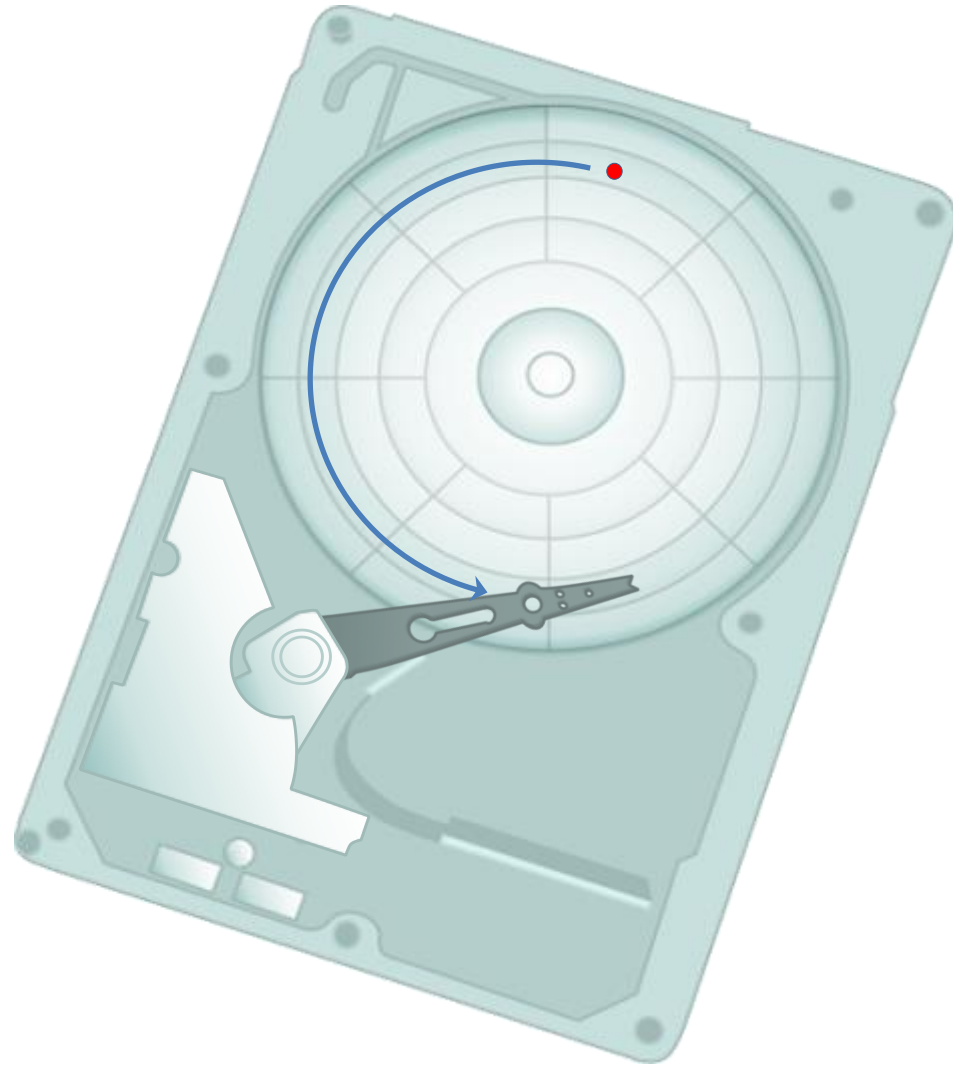
- Tempo gasto para reposicionar e pousar o braço e a cabeça na faixa correta
- Quanto menor o tempo de busca, mais rápida a operação de I/O
- Os fornecedores de disco informam as seguintes especificações de tempo de busca
 - Full stroke
 - Average
 - Track-to-track
- O tempo de busca de um disco é dado pelo fabricante



Latência Rotacional (*Rotation Latency*) $\overline{FI \wedge P}$

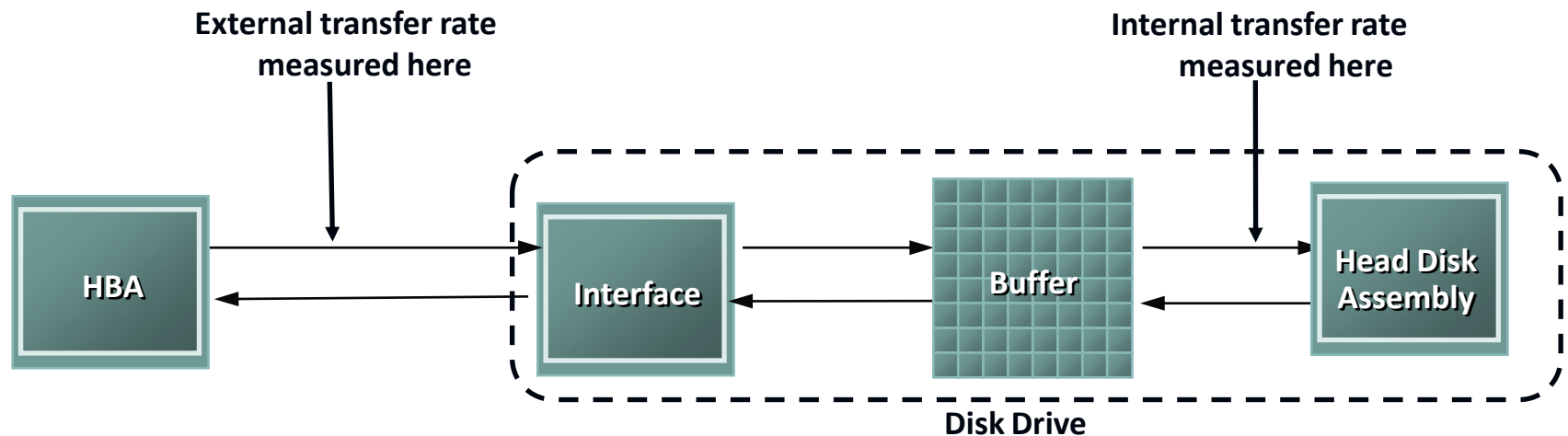
- O tempo gasto para o platter girar e posicionar os dados sob a cabeça de R/W
- Depende da velocidade de rotação do eixo
- Latencia rotacional média
 - Metade do tempo gasto para uma rotação completa
 - Para 'X' rpm, latencia rotacional é calculada em milisegundos pela fórmula:

$$= \frac{1/2}{(X/60)}$$



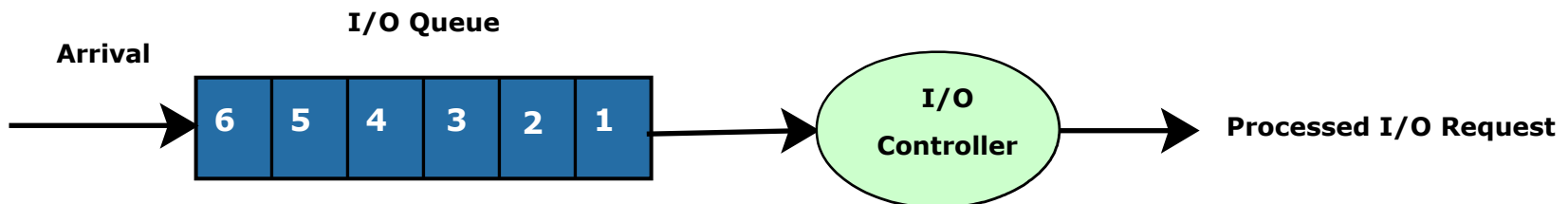
Taxa de Transferencia de Dados (*Data Transfer Rate*) $\overline{F} \wedge P$

- Volume médio de dados por unidade de tempo que o dispositivo pode transferir para o HBA
 - Taxa de transferencia interna : é a velocidade na qual os dados passam de uma única faixa de da superfície de um platter para o buffer (cache) interno do disco
 - Taxa de transferencia externa é a velocidade na qual os dados podem ser movidos através da interface para o HBA



Leis fundamentais que controlam o desempenho do disco

- Lei de Little
 - Descreve o relacionamento entre o número de solicitações em uma fila e o tempo de resposta
 - $N = a \times R$
 - “N” número total de solicitações no sistema da fila (solicitações na fila + solicitações no controlador de I/O)
 - “a” é a taxa de chegada, ou o número de solicitações de I/O que chegam ao sistema por unidade de tempo
 - “R” é o tempo de médio de resposta ou tempo de retorno de uma solicitação de I/O – o tempo total da chegada até a partida do sistema
- Lei da utilização
 - Define a utilização do controlador de I/O
 - $U = a \times R_s$
 - “U” é a utilização do controlador de I/O
 - “Rs” é o tempo de serviço, ou o tempo médio gasto por uma solicitação no controlador. $1/R_s$ é a taxa de serviço

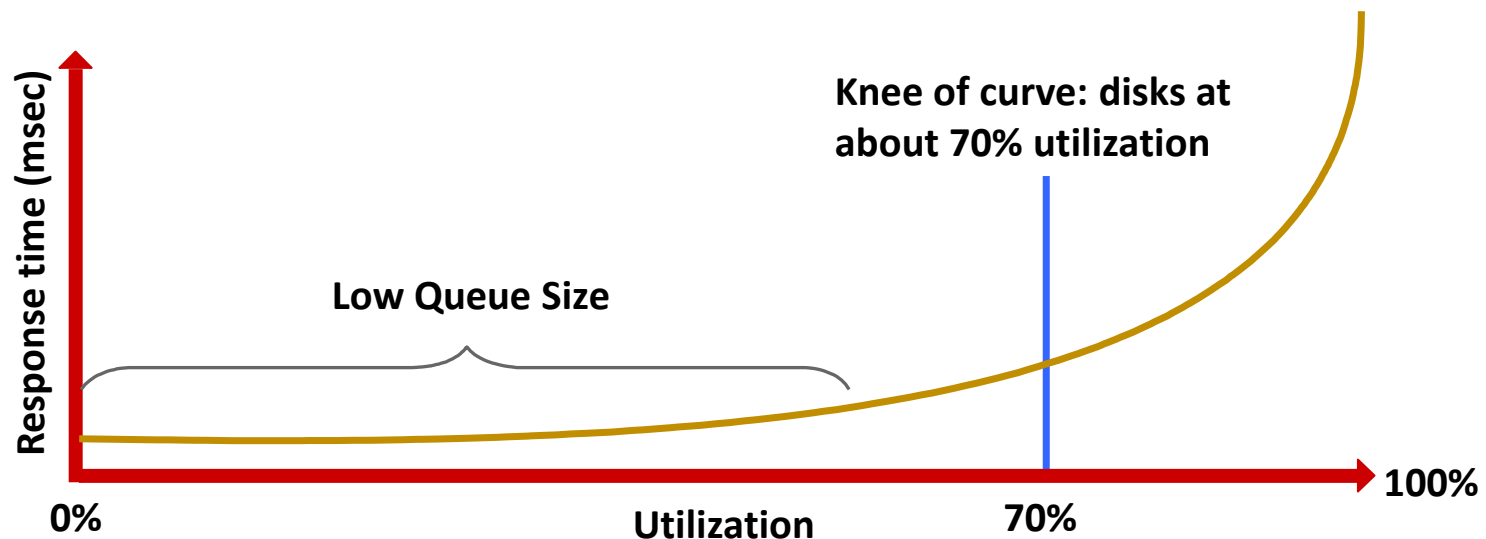


Utilização Versus Tempo de Resposta FIA/P

- Baseado na lei fundamental de desempenho do disco:

$$Av. Response Time = \frac{Service Time}{(1 - Utilization)}$$

- Service time é o tempo médio gasto por uma solicitação no controlador
- A fim de fornecer um tempo melhor de resposta os discos devem manter a menos de 70% de utilização.



- Discos necessários para atender a necessidade de capacidade de um aplicativo (D_C): $D_C = \frac{\text{Total capacity required}}{\text{Capacity of a single disk}}$
- Discos necessários para atender a necessidade de desempenho aplicação (D_P): $D_P = \frac{\text{IOPS generated by an application at peak workload}}{\text{IOPS serviced by single disk}}$
- IOPS fornecido por um disco (S) depende do tempo de serviço do disco (T_S): $T_S = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$
 - T_S é o tempo necessário para um I/O concluir, portanto IOPS fornecido por um disco (S) é igual a $(1/T_S)$
 - Para aplicações que necessitam de desempenho(S)= $0.7 \times \frac{1}{T_S}$

Discos necessários para uma aplicação = $\max(D_C, D_P)$

Exemplo

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Considere:

Ts – Tempo de serviço do disco

T – Tempo de Busca (*seek time*)

L – Latência Rotacional (*rotational latency*)

X – Tempo de Transferencia de Dados (*internal transfer time*)

$$T_s = T + L + X$$

Exemplo:

- O tempo médio de busca (*seek time*) em um ambiente de I/O aleatório é de 5 ms.
- Nesse caso T = 5.

Exemplo

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Considere:

T_s – Tempo de serviço do disco

T – Tempo de Busca (*seek time*)

L – Latência Rotacional (*rotational latency*)

X – Tempo de Transferencia de Dados (*internal transfer time*)

$$T_s = T + L + X$$

$$T_s = 5 + L + X$$

Exemplo:

- O tempo médio de busca (*seek time*) em um ambiente de I/O aleatório é de 5 ms.
- Nesse caso $T = 5$.

Exemplo

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Considere:

Ts – Tempo de serviço do disco

T – Tempo de Busca (*seek time*)

L – Latência Rotacional (*rotational latency*)

X – Tempo de Transferencia de Dados (*internal transfer time*)

$$T_s = T + L + X$$

$$T_s = 5 + 2 + X$$

Exemplo:

- A velocidade de rotação do disco é de 15000 RPM.
- Convertendo RPM para RPS: $15000/60 = 250$ RPS
- $L = 0.5 / 250 = 0.002$ segundos
- Mudando a unidade de segundo para milissegundo: $0.002 * 1000 = 2\text{ms}$

Exemplo

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Considere:

Ts – Tempo de serviço do disco

T – Tempo de Busca (*seek time*)

L – Latência Rotacional (*rotational latency*)

X – Tempo de Transferência de Dados (*internal transfer time*)

$$T_s = T + L + X$$

$$T_s = 5 + 2 + 0.8$$

Exemplo:

- Taxa de transferência interna é de 40 MB/s.
- Em um milissegundo serão transferidos 0,04 MB/ms.
- Tamanho do bloco é de 32KB. Convertendo para MB: $32\text{KB}/1024 = 0,03125 = 0,032\text{MB}$
- $X = 32\text{KB}/40\text{MB/s} = 0.032\text{MB}/0.04\text{MB/ms} = 0.8$

Exemplo

Considere:

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Ts – Tempo de serviço do disco

T – Tempo de Busca (*seek time*)

L – Latência Rotacional (*rotational latency*)

X – Tempo de Transferencia de Dados (*internal transfer time*)

$$T_s = T + L + X$$

$$T_s = 5 + 2 + 0.8$$

Exemplo:

- O tempo médio de busca (*seek time*) em um ambiente de I/O aleatório é de 5 ms logo, T = 5.
- A velocidade de rotação do disco é de 15000 RPM (ou 250 RPS) logo, L = 0.5/250
- O tempo de transferência é de 40MB/s e o tamanho do bloco é de 32KB logo, X = 32KB/40MB/s = 0.032MB/0.04MB/ms = 0.8
- Ts = 5 + 2 + 0.8 = **7.8**

Exemplo de problema:

- Considere uma aplicação que precise de 1TB de capacidade de armazenamento execute 4900 IOPS
 - O tamanho do I/O (I/O size) da aplicação é 4KB
 - Como se trata de aplicação crítica para o negócios, o tempo de resposta deve estar dentro de intervalo aceitável
- Especificação de unidade de disco disponível:
 - Capacidade da unidade de disco = 73 GB
 - 15000 RPM
 - 5 ms tempo médio de busca (average seek time)
 - Taxa de transferência 40 MB/sec

Calcule o numero de discos necessários?

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Solução

- Calcule o tempo necessário para executar um I/O

=Seek time + (rotational delay/speed in RPM)+ (block size/transfer rate)

Portanto, 5 ms + (0.5/15000) + 4KB/(40MB/s) =

5 ms + (0.5/(1500/60) + 0.004MB/(0,04MB/ms) =

5 ms + (0.5/250) + 0,1 ms =

5 ms + 0,002 s + 0,1 ms =

5 ms + 2 ms + 0,1 ms = 7.1 milissegundos

$$T_s = \text{Seek time} + \frac{0.5}{(\text{Disk rpm}/60)} + \frac{\text{Data block size}}{\text{Data transfer rate}}$$

Solução

- Calcule o tempo necessário para executar um I/O
= Seek time + (rotational delay/speed in RPM) + (block size/transfer rate)
Portanto, 5 ms + (0.5/15000) + 4KB/(40MB/s) = 7.1 ms
- T_s é o tempo necessário para um I/O concluir, portanto IOPS fornecido por um disco (S) é igual a $(1/T_s)$
- Calcule o número máximo de IOPS que o disco pode executar
 - $1 / T_s = 1 / 7.1 \text{ ms} = 0,140845 \text{ I/O por milissegundo}$
 - $0,140845 * 1000 = 140 \text{ IOPS}$

Solução

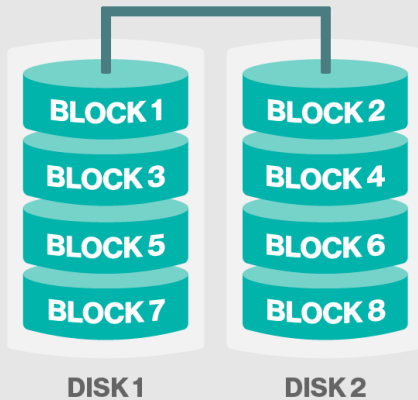
- Calcule o tempo necessário para executar um I/O
= Seek time + (rotational delay/speed in RPM) + (block size/transfer rate)
Portanto, $5 \text{ ms} + (0.5/15000) + 4\text{KB}/(40\text{MB/s}) = 7.1 \text{ msec}$
- Calcule o número máximo de IOPS que o disco pode executar
 - $1 / 7.1 \text{ ms} = 140 \text{ IOPS}$
- Por razões de desempenho a utilização deve ser menor que 70%
 - Portanto, $140 \times 0.7 = 98 \text{ IOPS}$
- A aplicação irá precisar de:
 - $4900/98$ i.e. 50 disk, para requisitos de performance
 - $1\text{TB} / 73 \text{ GB}$ i.e. 14 disk para requisitos de capacidade

Disk required = max (capacity, performance)

RAID (Redundant Array of Independent Disks)

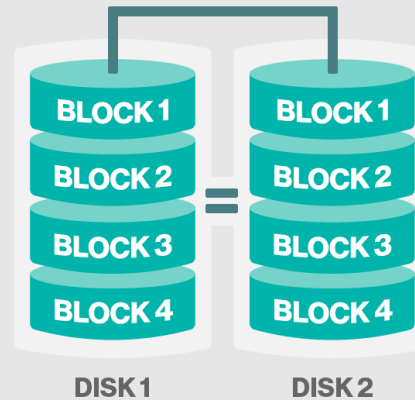
RAID 0

Striping

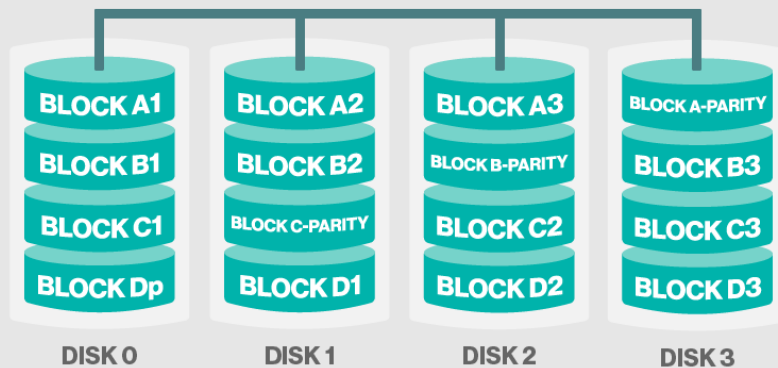


RAID 1

Mirroring

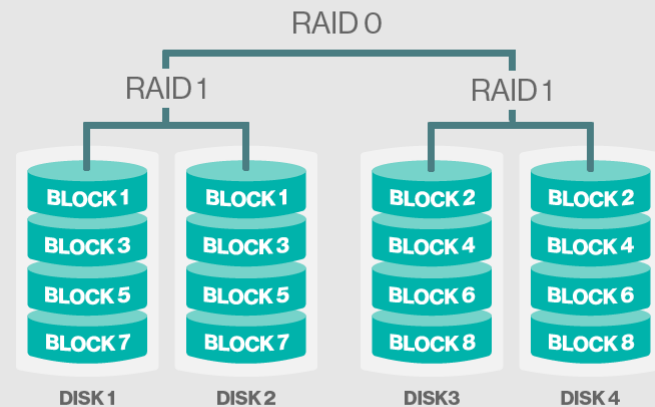


RAID 5



RAID 10 (RAID 1+0)

Stripe + Mirror



Um problema para Big Data

- Logística
- O sistema de segurança envia a posição de cada container a cada cinco minutos junto com uma imagem da carga.
- O arquivo pode ter até 1 TB
- A carga foi desviada?

```
145,4155924;0,047417174;20,0061;17,931;16,86136;16,40727;16,07867;536102941950502000;0,1
145,3514537;-1,23853648;19,41876;18,08052;17,46687;17,08977;16,86917;299568842282133000;
145,1866721;-0,89339385;19,50078;18,34422;17,72999;17,33761;17,11797;535930318624942000;
145,2138366;-0,322982364;20,13664;18,1309;17,06312;16,60838;16,25545;535934716671453000;
145,2693476;-0,470419246;21,14657;18,97346;17,78584;17,3233;16,98022;535934441793546000;
145,3451455;1,087684363;19,49029;17,63432;16,63974;16,18519;15,78099;537066109304596000;
145,2452786;0,266996641;19,56595;17,40511;16,37076;15,88421;15,49268;536100742927247000;
145,5544488;0,223322017;20,31785;18,89908;17,8659;17,31925;16,94901;299595505439107000;0
145,8925224;1,117732004;19,12206;17,3172;16,55206;16,0947;15,73396;540505106177615000;0,
145,7887508;0,51945584;19,62373;18,36254;17,79043;17,36876;17,2051;540513077636917000;0,
146,246995;1,115306497;20,55324;18,84997;17,67538;17,08179;16,64676;540493286427617000;0
146,1504517;0,820095273;18,71381;17,45835;16,7241;16,26816;15,99526;540491912038082000;0
146,7729359;0,701144998;19,55638;18,27996;17,79401;17,57318;17,30269;300710959616387000;
148,6506582;-1,035642206;20,72381;18,33205;17,09071;16,57671;16,21045;301823665803126000
```

Tempo de Execução

- Tempo de acesso aos dados +
- Tempo de processamento (~60 min) +
- Tempo de transmissão dos dados
> 180 min

Tempo médio de acesso aos dados – 122 MB/seg.
Um arquivo de 1 TB será lido em 2 horas e meia!

1 segundo – 122 MB

X segundo – 1048576 MB

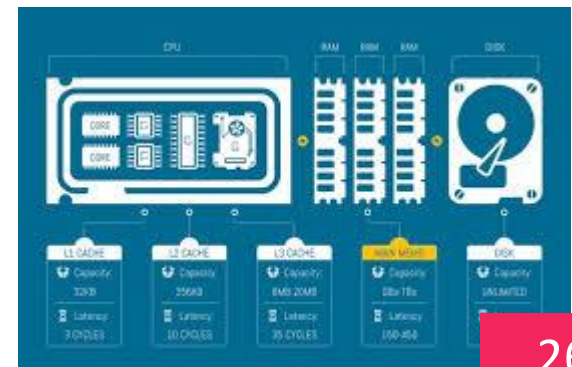
$$X = 1048576 / 122 = 8585 \text{ segundos}$$

$$X = 2\text{h}22$$

Onde está o “gargalo”?

- Velocidade das CPUs esta cada vez maior mas...
- A velocidade de acesso a disco, ou volumes de discos ainda é lenta.
- O aumento da velocidade de CPU não beneficia muito os programas que tem necessidade de acessar grandes volumes de dados.

Fonte: Oracle



Hadoop – Uma boa solução ^{FIA/P}



O que é?

- Diferentes públicos, diferentes definições:
 - Executivos: “é um projeto de software livre da Apache que tem como objetivo obter valor do volume/velocidade/variedade incrível de dados sobre sua organização. Use os dados em vez de jogar a maioria fora.”;
 - Gerentes Técnicos: “um conjunto de softwares livres que minera o BigData estruturado e não estruturado de sua empresa. Ele integra com seu ecossistema existente de Business Intelligence”;
 - Jurídico: “um conjunto de software livre empacotado e suportado por diversos fornecedores. Consulte a seção Recursos relacionada à indenização de IP”;

O que é?

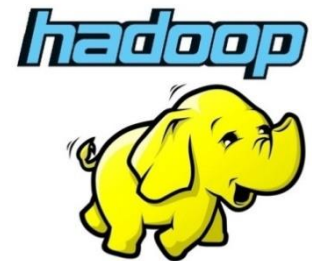
- Diferentes públicos, diferentes definições:
 - Engenharia: “um ambiente de execução Mapear/Reduzir massivamente paralelo, sem compartilhamento e baseado em Java. Pense em centenas a milhares de computadores trabalhando no mesmo problema, com resiliência integrada contra falhas. Projetos no ecossistema Hadoop fornecem carregamento de dados, linguagens de nível superior, implementação automatizada na nuvem e outros recursos”;
 - Segurança: “um suite de software protegido por Kerberos”. Nota: “Kerberos é o nome de um Protocolo de rede, que permite comunicações individuais seguras e identificadas, em uma rede insegura”.

Hadoop

- Projeto código aberto mantido pela Apache Foundation.
- Fornece uma implementação de código aberto do modelo de programação *MapReduce* de forma confiável e escalável.
- Projetado para ampliar o processamento de um único servidor em milhares de máquinas, onde cada uma das máquinas oferecem poder de processamento e armazenamento local.
- Esta ferramenta é utilizada para processamento em *batch* de grandes volumes de dados (*Big Data*).

Hadoop – Uma boa solução FIAP

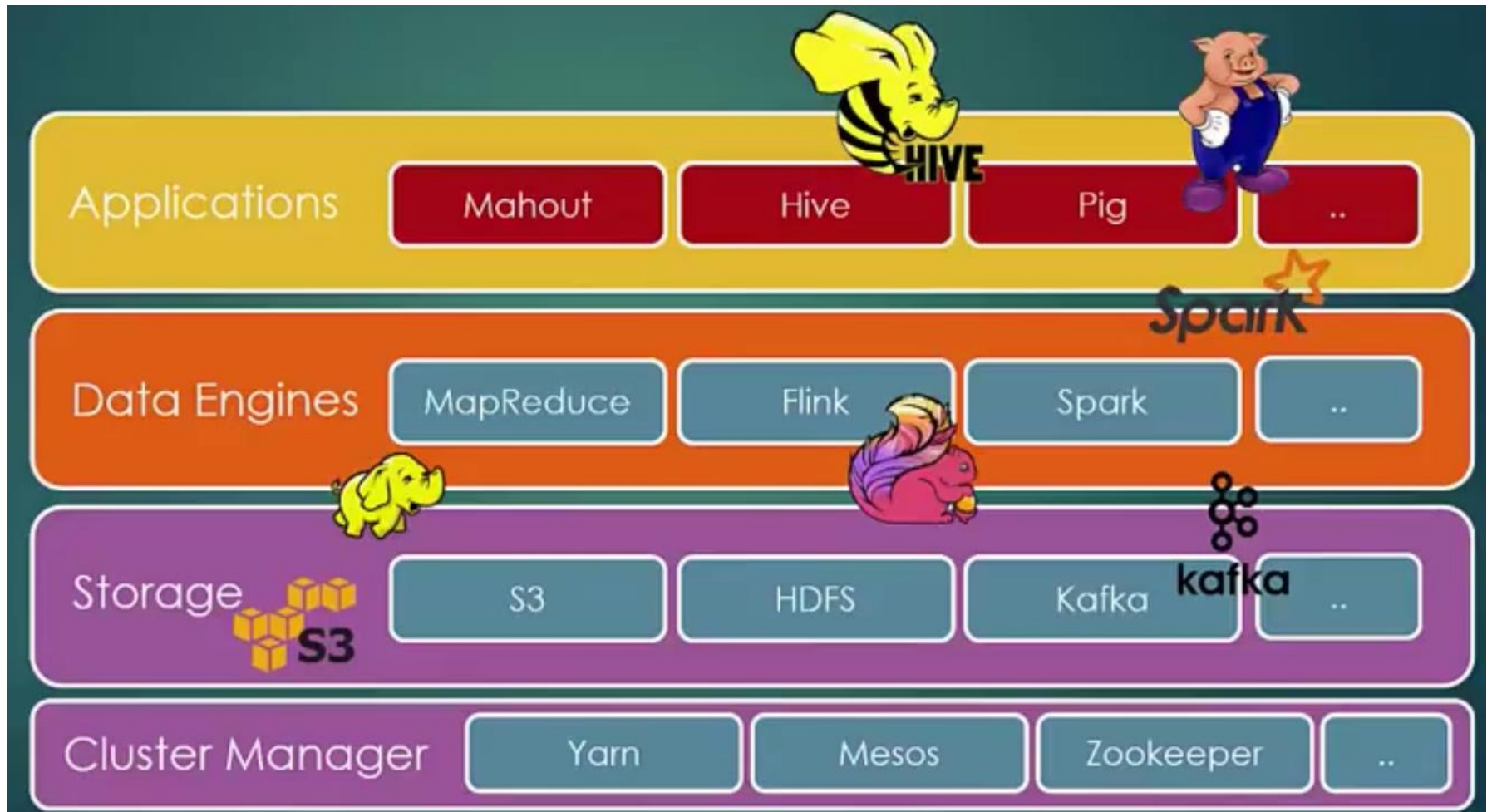
- Suporta grande volume de dados
- Armazenamento eficiente
- Boa solução de recuperação de dados
- Escalabilidade Horizontal
- Bom custo/benefício
- Simples para programadores e não-programadores

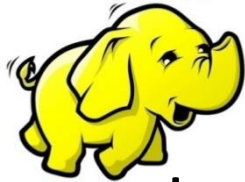


Hadoop – Uma boa solução, mas não é a única!



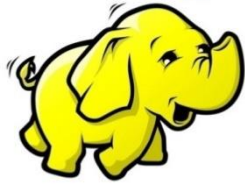
Panorama do Big Data Apache





Hadoop

- Segundo a Hadoop, “Hadoop é um *storage* confiável e um sistema analítico” [2014]
- Composto por duas partes essenciais:
 - o Hadoop Distributed Filesystem (HDFS), sistema de arquivos distribuído e confiável, responsável pelo armazenamento dos dados
 - Hadoop MapReduce, responsável pela análise e processamento dos dados.
- O nome do projeto veio do elefante de pelúcia que pertencia ao filho do criador, Doug Cutting.



Hadoop

- Uma das alternativas para Big Data
 - Hadoop (1.x ou 2.x)
- Open Source
- Suporta várias API's (Pig/Hive/Spark)
- Multiplataforma (desenvolvido em Java)
- Escalável
- Criado em 2006 (Nutch Project)
 - MapReduce introduzido em 2004
 - Hadoop Distributed File System (HDFS) introduzido em 2007.
 - Primeira distribuição oficial: 1.0.0 (2011)

Hadoop 1 vs Hadoop 2

Single Use System

Batch Apps

HADOOP 1.0

MapReduce

(cluster resource management
& data processing)

HDFS

(redundant, reliable storage)



Multi Purpose Platform

Batch, Interactive, Online, Streaming, ...

HADOOP 2.0

MapReduce

(data processing)

Others

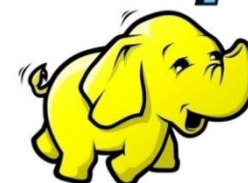
YARN

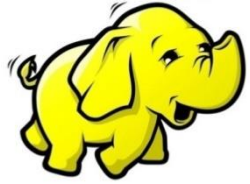
(cluster resource management)

HDFS2

(redundant, highly-available & reliable storage)

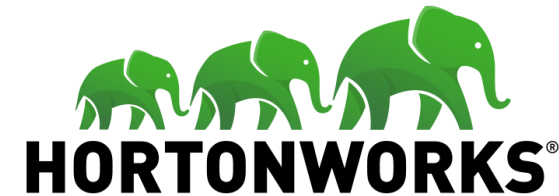
hadoop



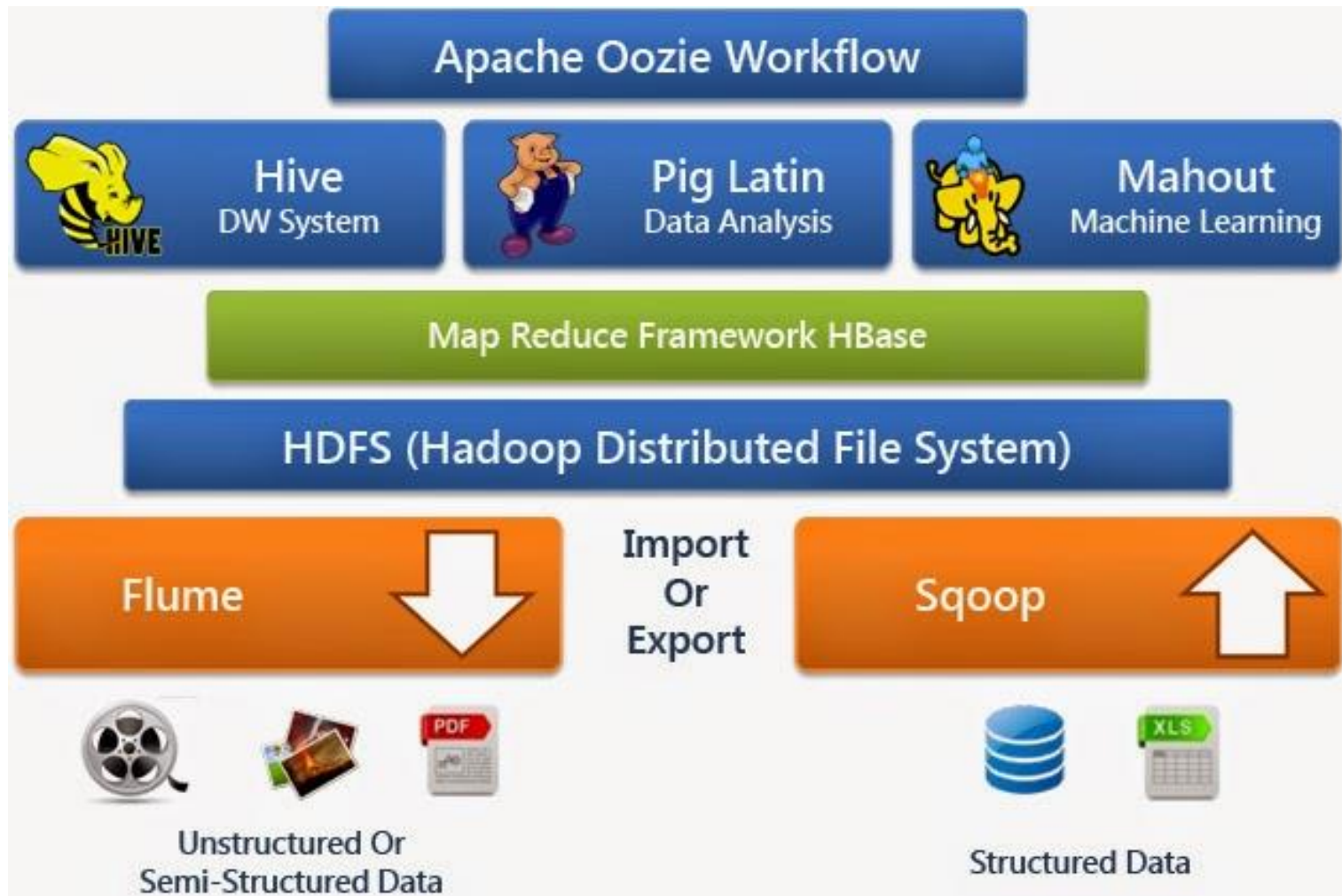


Versões Comerciais

- HortonWorks
- MapR
- Cloudera
- Microsoft HDInsight for Hadoop (HortonWorks)
- Amazon Elastic Map Reduce (EMR)



Ecosistema Hadoop



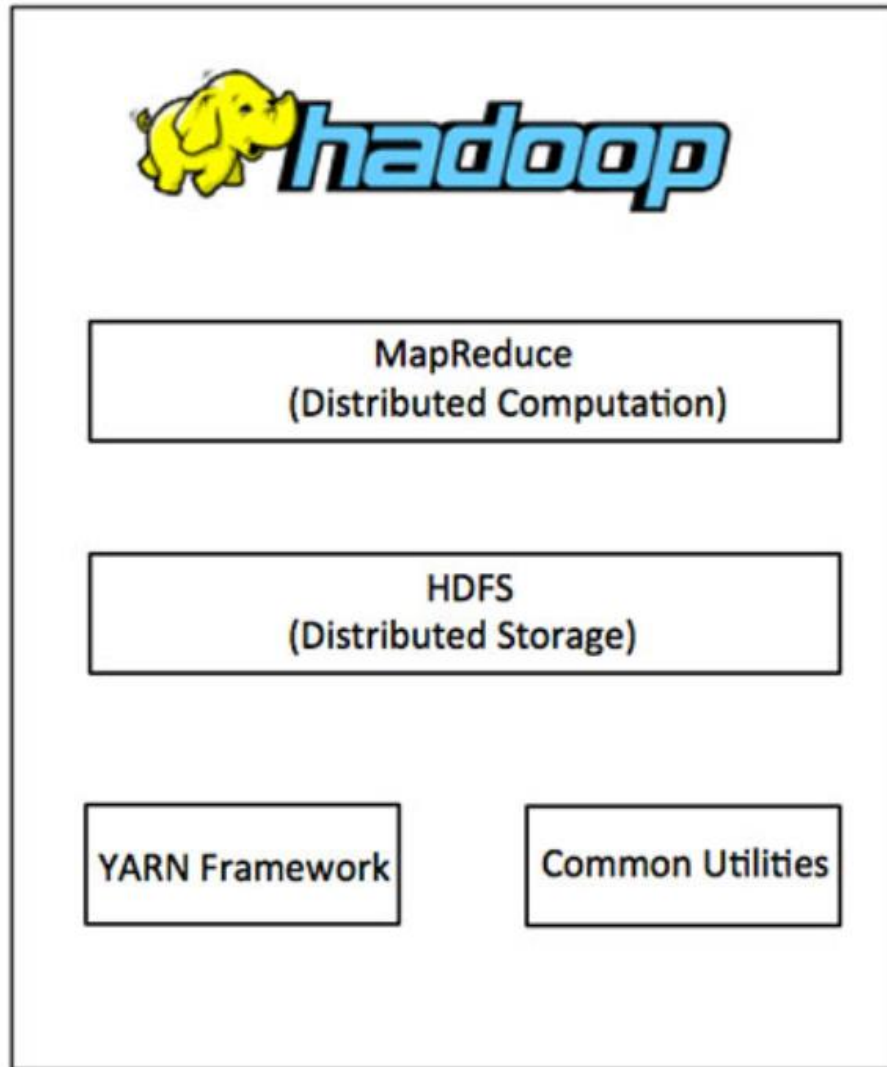
Componentes do Hadoop

- O *framework* Hadoop é composto pelos seguintes módulos:
 - **Hadoop Common**: composto por bibliotecas JAVA e utilitários necessários para outros módulos. As bibliotecas provem abstração no nível do Sistema Operacional e sistema de arquivos e também possuem todos os arquivos e scripts necessários para inicializar o Hadoop;

Componentes do Hadoop

- Hadoop YARN: *framework* para agendamento de tarefas (*jobs*) e gerenciamento de recursos do *cluster*;
- Hadoop *Distributed File System (HDFS)*: sistema de arquivos distribuído que fornece acesso aos dados com elevadas taxas de transferência;
- Hadoop MapReduce: sistema baseado no Hadoop YARN para processamento paralelo de grandes volume de dados.

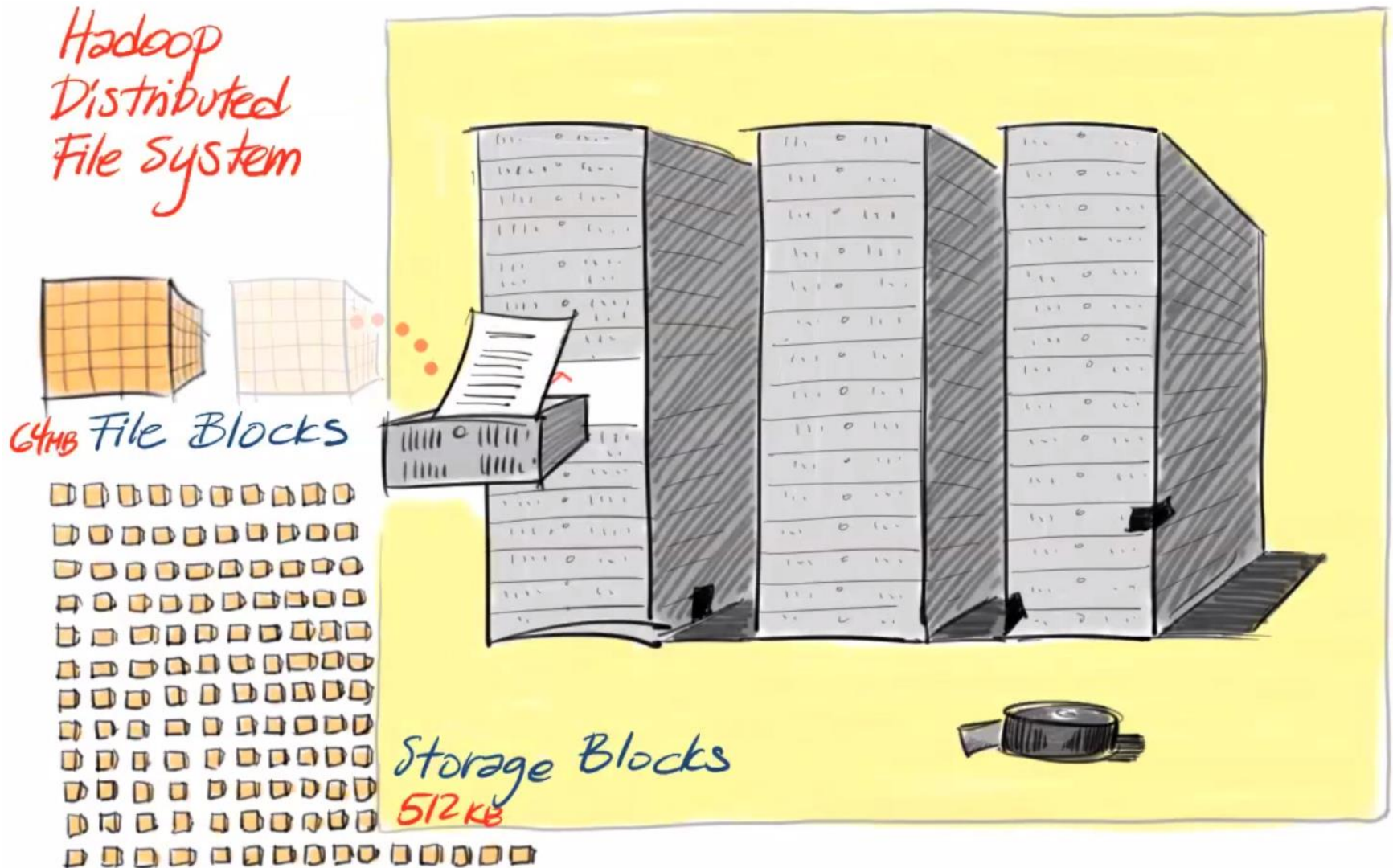
Diagrama Básico dos Componentes



Fonte: http://www.tutorialspoint.com/hadoop/hadoop_introduction.htm

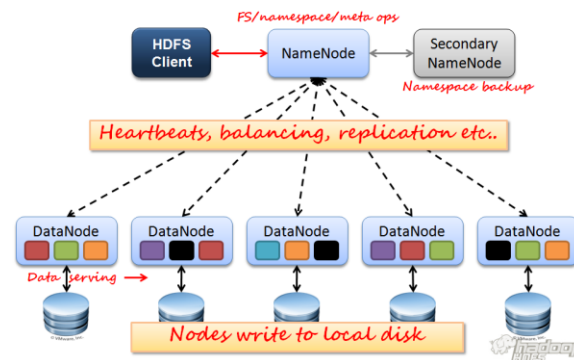
HDFS

FIAP



HDFS

- O *Hadoop Distributed File System* (HDFS) é um sistema de arquivos altamente tolerante a falhas projetado para executar em hardware padrão de baixo custo.
- O HDFS disponibiliza acesso de alto rendimento para os dados do aplicativo e é adequado para aplicativos com grandes conjuntos de dados.



HDFS

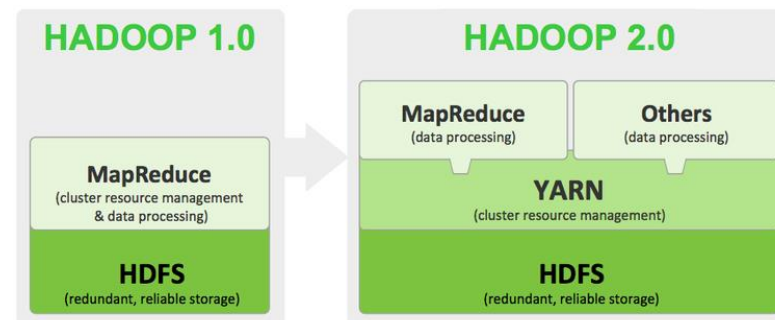
- É possível utilizar o Hadoop com vários tipos de sistemas de arquivos como sistema de arquivo local, HFTP, S3 e outros, no entanto o sistema de arquivos normalmente utilizado é o HDFS;
- Utiliza arquitetura *Master/Slave*, onde o *Master* é um único **NameNode** que gerencia os metadados do sistema de arquivo, enquanto os **DataNodes** armazenam os dados em si;

HDFS

- O HDFS é baseado no GFS (Google *File System*) e provê um sistema de arquivo distribuído que foi projetado para ser executado tanto em sistema de *cluster* gigantescos (milhares de computadores) quanto em pequenos sistemas com poucos computadores de uma maneira confiável e tolerante a falhas;

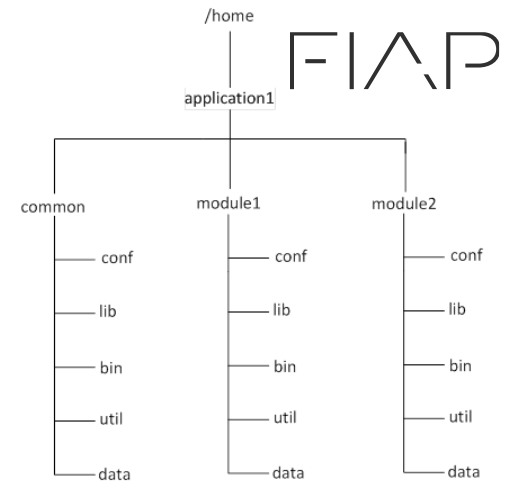
HDFS

- Redundante
- Armazenamento Confiável
- Persiste o dado de forma distribuída
- Pode usar hardware de propósito genérico
- Executa no sistema de arquivos local
- Tolerante a falha
- Arquivos são fragmentados em blocos (chunks)
- Write Once, Read Many (WORM)



HDFS

- HDFS usa uma arquitetura master/slavor:
 - (1) Namenode: Data manager
 - (N) Datanode: Data Storage
- Sistema de arquivo baseado em diretórios
 - Operações disponíveis em HDFS:
 - Create, Remove, Read, Write
 - Essas operações são gerenciados pela Namenode



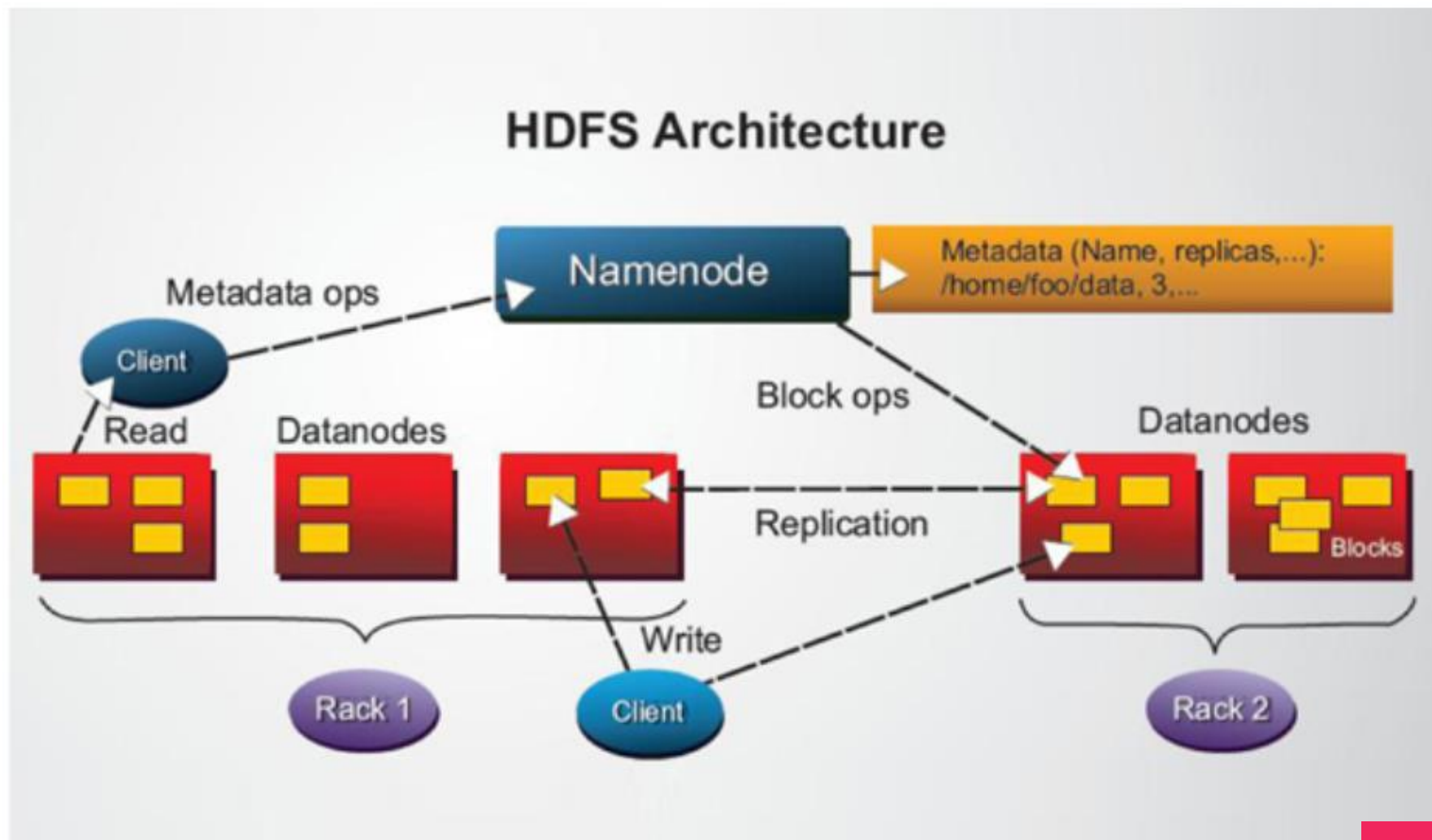
HDFS

- Dados são particionados em chunks
 - O tamanho do chunk pode ser parametrizado na instalação do Hadoop. Por padrão: 64 MB
 - Exemplo: Um arquivo de 256 MB será dividido em quatro chunks de 64 MB.
 - Um arquivo no HDFS é separado em vários chunks e estes chunks são armazenados nos DataNodes;

HDFS

- O Namenode é responsável pela replicação dos dados
 - Faz mapeamento dos chunks nos datanodes
 - Usa “heartbeat” para controlar a “saúde” dos dados.
 - Executa balanceamento de dados, se necessário
- Datanode fornece, apenas, armazenamento para os dados.
 - Cria, exclui, e replica os chunks conforme as orientações do namenode.

HDFS



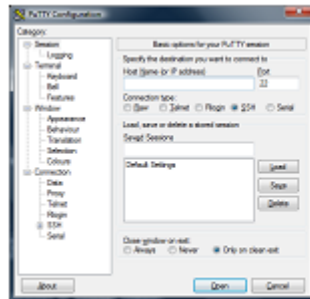
Acesso ao Cluster Hadoop

- Linux

```
ssh -i ~/cluster_key/hirwuser150430.pem  
hirwuser150430@54.85.143.224
```

Preparando o Ambiente

- Descompacte o arquivo cluster_key.zip.
- <http://www.putty.org/>



Download PuTTY

PuTTY is an SSH and telnet client, developed source code and is developed and supported

You can download PuTTY [here](#).




You probably want one of these. They include all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

MSI ('Windows Installer')

32-bit: [putty-0.70-installer.msi](#) (or by FTP) (signature)

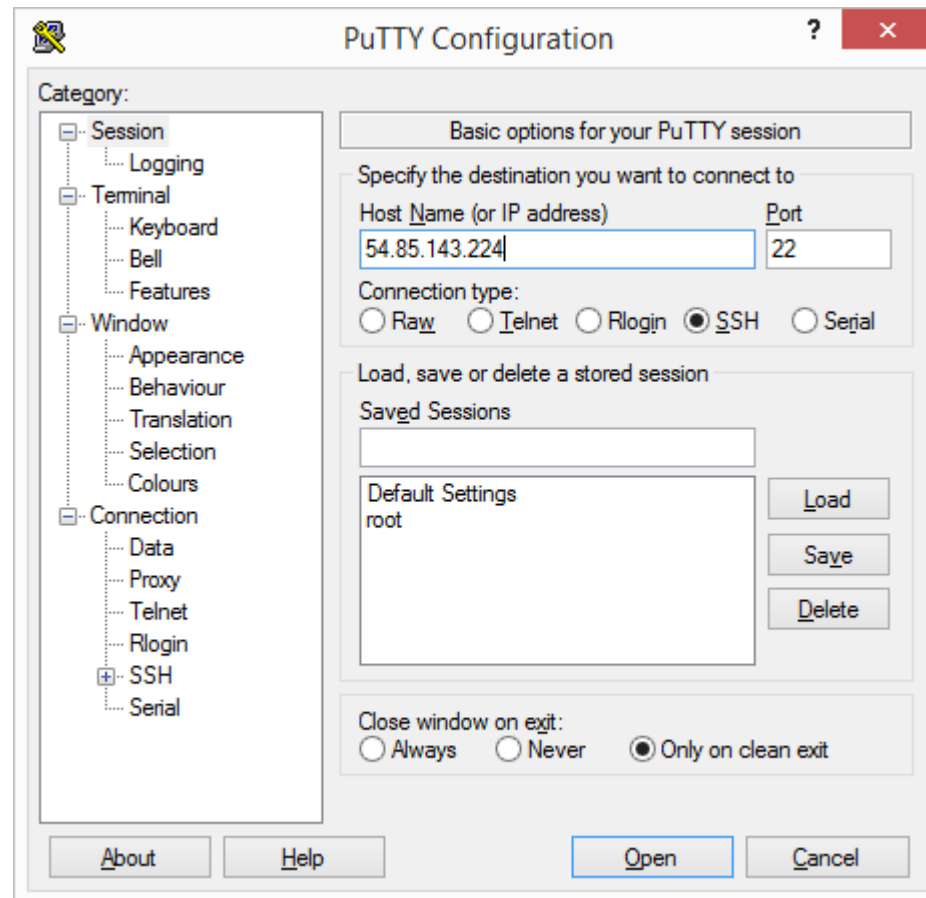
64-bit:  [putty-64bit-0.70-installer.msi](#) (or by FTP) (signature)

Unix source archive

.tar.gz: [putty-0.70.tar.gz](#) (or by FTP) (signature)

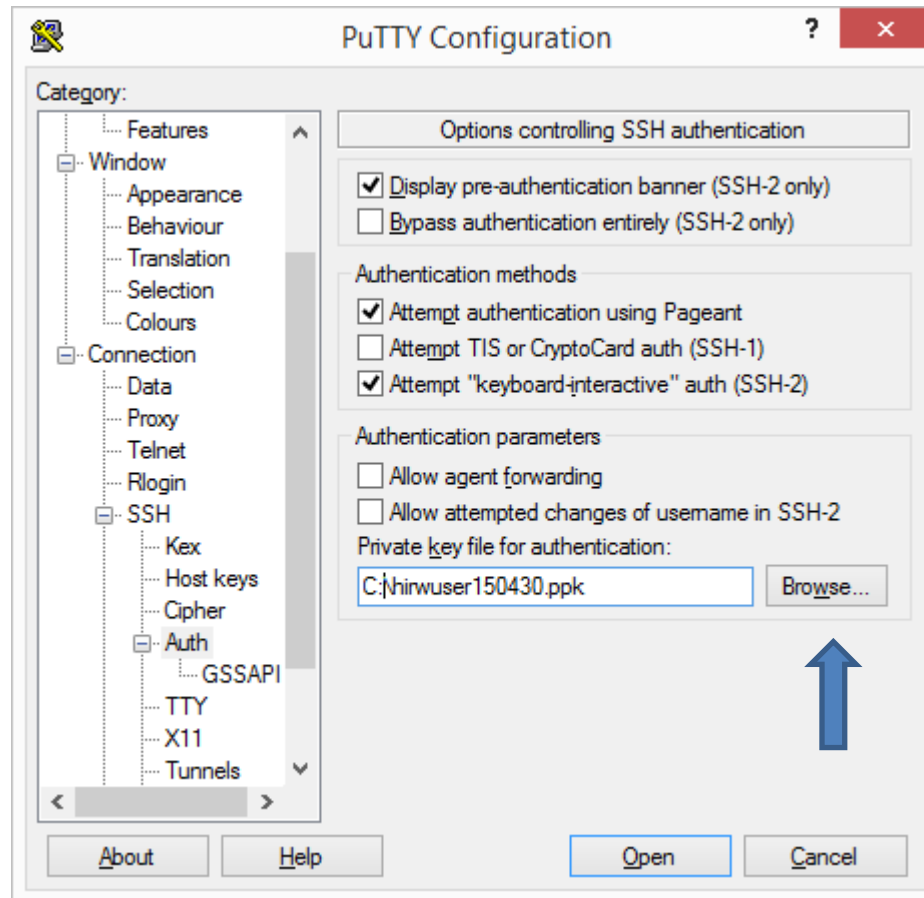
Preparando o Ambiente

- Em Host Name, digite: 54.85.143.224



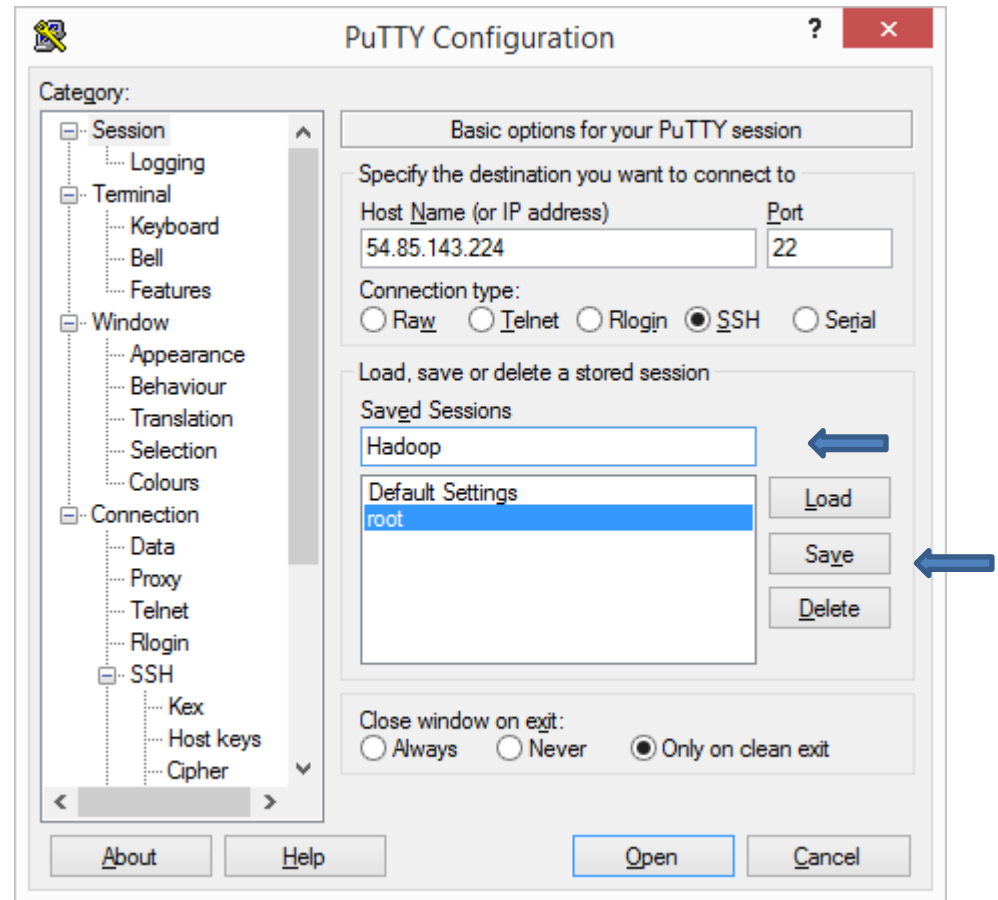
Preparando o Ambiente

- Em Connection -> SSH -> Auth



Preparando o Ambiente

- Em Session
- Defina um nome
- Salve
- Clique em Open



- Entre com o usuário

- `hirwuser150430`

- Listar o diretório root

- `hadoop fs -ls /`

- Listar o diretório home

- `hadoop fs -ls`

- `hadoop fs -ls /user/hirwuser150430`



```
login as: hirwuser150430
```



Comandos para o S.O. local

- ls – lista o diretório
 - ls
 - ls > teste.txt
- cat – lista o conteúdo de um arquivo
 - cat teste.txt
- mkdir – cria diretório
 - mkdir teste

Comandos para o S.O. local

- cp – copia arquivo
 - cp teste.txt teste1.txt
- mv – move um arquivo
 - mv teste1.txt /etc/teste2.txt
- rm – apaga arquivo
 - rm /etc/teste2.txt

Comandos Frequentemente Usados FIAP

- Comandos *shell* são usados para executar várias operações Hadoop HDFS e para gerenciar os arquivos presentes em clusters HDFS.
- Todos os comandos são invocados pelo script `/bin/hdfs` ou `/bin/hadoop`.
- `version`.
 - Imprime a versão do hadoop
 - `hadoop version`

Comandos Frequentemente Usados FIAP

- `classpath`.
 - Exibe o caminho das classes e bibliotecas do Hadoop. Definido no arquivo `hadoop-config.sh`.
 - `hadoop classpath`

- `getconf.`
 - Exibe informações de configuração do diretório de configurações.
 - `hdfs getconf -namenodes`
 - `hdfs getconf -secondaryNameNodes`
 - `hdfs getconf -backupNodes`
 - `hdfs getconf -includeFile`
 - `hdfs getconf -excludeFile`
 - `hdfs getconf -nnRpcAddresses`

Comandos Frequentemente Usados FIAP

- **mkdir.**
 - Cria um diretório no *path* informado.
 - `hadoop fs -mkdir teste1-seunome`
 - `hadoop fs -mkdir teste2-seunome`
 - `hadoop fs -mkdir teste3-seunome`

Comandos Frequentemente Usados FIAP

- `ls`.
 - Exibe uma lista do conteúdo de um diretório especificado no *path*. Pode exibir permissões, proprietário, tamanho e data de criação/alteração de cada arquivo.
 - `hadoop fs -ls`
 - `hadoop fs -ls -R`

Comandos Frequentemente Usados FIAP

- `copyFromLocal`.
 - Copia um arquivo ou diretório do *file system* local para o destino especificado no HDFS.
 - `hadoop fs -copyFromLocal /hmrw-starterkit/hdfs/commands/dwp-payments-april10.csv teste1-seunome`

Comandos Frequentemente Usados FIAP

- **cat.**
 - Exibe o conteúdo de um arquivo na console ou na **stdout**.
 - `hadoop fs -cat teste1-seunome/dwp-payments-april10.csv`

Comandos Frequentemente Usados FIAP

- `copyToLocal`.
 - Copia um arquivo ou diretório do HDFS para o *file system* local.
 - `hadoop fs -copyToLocal teste1-seunome/dwp-payments-april10.csv .`
 - `hadoop fs -ls teste1-seunome`

Comandos Frequentemente Usados FIAP

- `cp`
 - Copia um arquivo ou diretório de uma origem identificada no comando para um destino identificado no comando
 - `hadoop fs -cp teste1-seunome/dwp-payments-april10.csv teste2-seunome`

Comandos Frequentemente Usados FIAP

- mv
 - Move um arquivo ou diretório de uma origem identificada no comando para um destino identificado no comando
 - `hadoop fs -mv teste1-seunome/dwp-payments-april10.csv teste3-seunome`

Comandos Frequentemente Usados FIAP

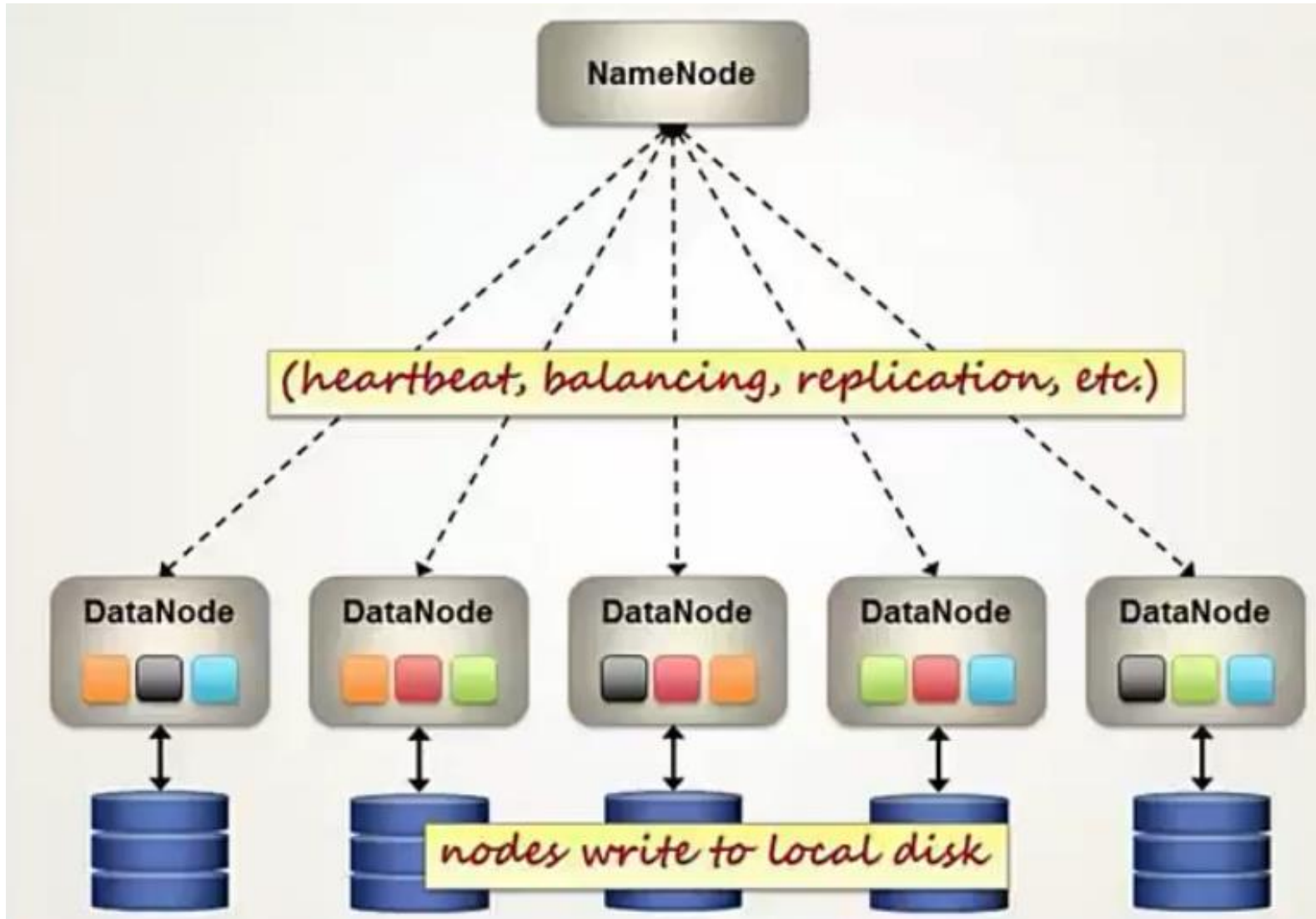
- **chmod**
 - Altera as permissões de leitura, gravação e execução de um arquivo
 - `hadoop fs -chmod 777 teste2-seunome/dwp-payments-april10.csv`

	Owner	Group	Other
Read (r)	4	4	4
Write (w)	2	2	2
Execute (x)	1	1	1
Total	7	7	7

Comandos Frequentemente Usados FIAP

- **rm**
 - Apaga um arquivo ou diretório
 - `hadoop fs -rm teste2-seunome/dwp-payments-april10.csv`
 - `hadoop fs -rm -r teste1-seunome`
 - `hadoop fs -rm -r teste2-seunome`
 - `hadoop fs -rm -r teste3-seunome`

HDFS



Namespace

- “É um delimitador abstrato (container) que fornece um contexto para os itens que ele armazena (nomes, termos técnicos, conceitos), o que permite uma desambiguação para itens que possuem o mesmo nome mas que residem em espaços de nomes diferentes. Como um contexto distinto é fornecido para cada container, o significado de um nome pode variar de acordo com o espaço de nomes o qual ele pertence” (O’Reilly Strata, 2015)

NameNode

- Conhecido do como nó mestre (*master node*)
- É o servidor que armazena os metadados da árvore de diretório, réplicas, número de bloco de dados e outros detalhes
- Os metadados estão disponíveis na memória para recuperação mais rápida dos dados.

NameNode

- Mantêm e gerencia os nós escravos e atribui tarefas a eles.
- Normalmente implementado em hardware confiável pois é peça central do HDFS.

NameNode

```
cd /etc/hadoop/conf  
ls -ltr  
vi core-site.xml
```

- A **property fs.defaultFS** especifica a localização do **namenode**.

```
<property>  
  <name>fs.defaultFS</name>  
  <value>hdfs://ip-172-31-45-216.ec2.internal:8020</value>  
</property>
```

- Este arquivo fica disponível em todos o **nodes** do **cluster** permitindo que todos **nodes** saibam a localização do **namenode**.

Tarefas do NameNode

- Gerencia o *namespace* do sistema de arquivos
- Regula o acesso do cliente aos arquivos
 - Não existe quota de usuário
- Executa operações do *file system*
 - nomeação, abertura e fechamento de arquivos ou diretórios.

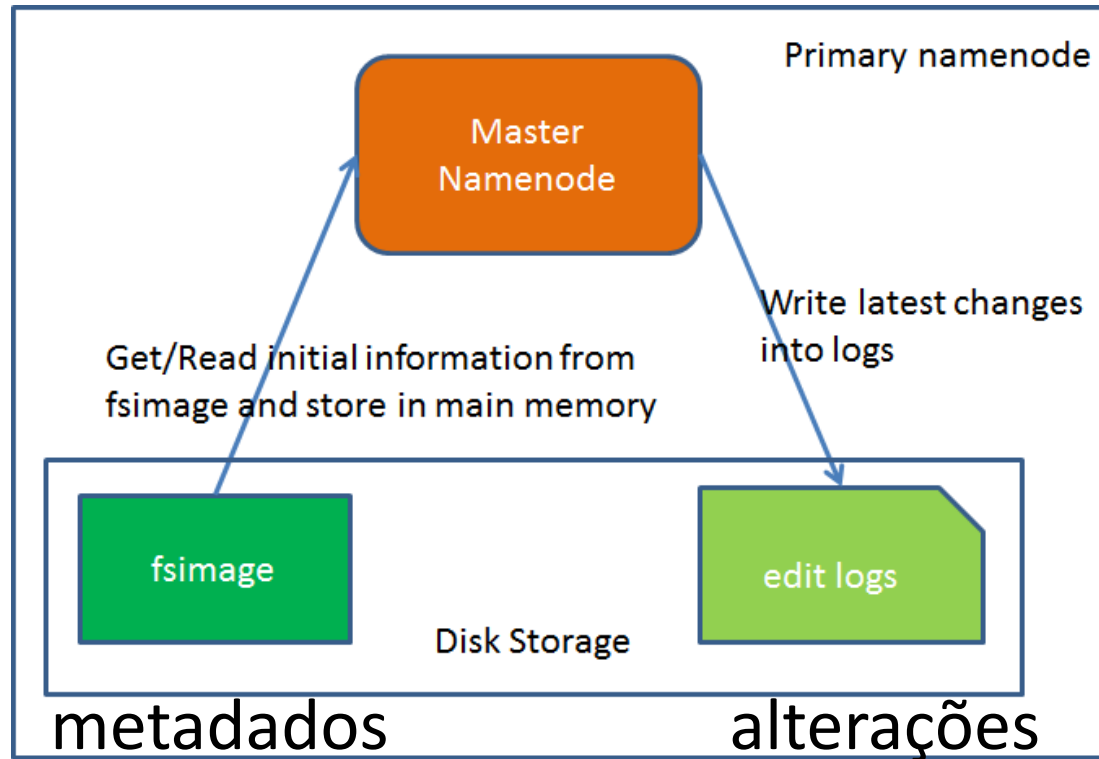
Tarefas do NameNode

- Todos os DataNodes enviam *Heartbeats* e *block reports* para o NameNode.
 - *Heartbeats* garantem que o nó está ativo
 - *Block Report* contém a lista de blocos do datanode
- Responsável por cuidar do *Replicator Factor* de todos os blocos.

Arquivos do NameNode

- Arquivos presentes nos metadados do NameNode:
- **FsImage**
 - É um "arquivo de imagem".
 - Contém o *namespace* inteiro do sistema de arquivos
 - Armazenado como um arquivo no file system no namenode
 - Contém uma forma serializada de todos os diretórios e arquivos *inode* do sistema
 - **Inode** é uma representação interna dos metadados dos arquivos e diretórios.
- **EditLogs**
 - Contém todas as alterações recentes feitas no **FsImage** mais recente.
 - Ao receber uma solicitação de criação, atualização ou exclusão do cliente, o Namenode primeiro registra essa solicitação no arquivo de edição.

Arquivos do NameNode



DataNode

- Conhecido como *Slave*.
- Armazena dados reais no HDFS.
- Executa a operação de leitura e gravação de acordo com a solicitação do cliente.
- DataNodes podem ser implantados em hardware de commodities.

NameNode

```
cd /etc/hadoop/conf  
ls -ltr  
vi hdfs-site.xml
```

- **dfs.namenode.name.dir** especifica onde o **NameNode** pode armazenar seus arquivos, localmente.
- **dfs.datanode.name.dir** especifica onde o **DataNode** pode armazenar arquivos e blocos, localmente
- **dfs.namenode.http.address** fornece o endereço no **NameNode**. O endereço pode ser acessado através de um *browser* e fornecer informações sobre o HDFS.

```
<property>  
  <name>dfs.namenode.http-address</name>  
  <value>ec2-54-92-244-237.compute-1.amazonaws.com:50070</value>
```

Tarefas do DataNode

- Bloquear a criação, exclusão e replicação conforme as instruções enviadas pelo **NameNode**.
- Gerenciar o armazenamento de dados do sistema.
- Enviar **heartbeats** para o **NameNode**
 - Por padrão, a frequência é de 3 segundos.

Metadata Disk Failure

- **FsImage** e **EditLog** são estruturas de dados centrais do HDFS.
- A corrupção desses arquivos pode fazer com que a instância do HDFS não funcione.
- O NameNode pode ser configurado para suportar várias cópias do FsImage e EditLog.
- As cópias são atualizadas de forma síncrona.
- A atualização síncrona de múltiplas cópias do **FsImage** e **EditLog** pode degradar a taxa de transações por segundo do NameNode.

Metadata Disk Failure

- Aplicações HDFS são muito intensivas em dados mas não são intensivas em metadados.
- Quando um NameNode é reiniciado, seleciona o **FsImage** e **EditLog** mais recente e consistente para usar.
- Também é possível habilitar Alta Disponibilidade usando múltiplos NameNodes com armazenamento compartilhado no NFS (*Network File System*) ou usando um recurso de distribuição denominado **Journal**(recomendado).

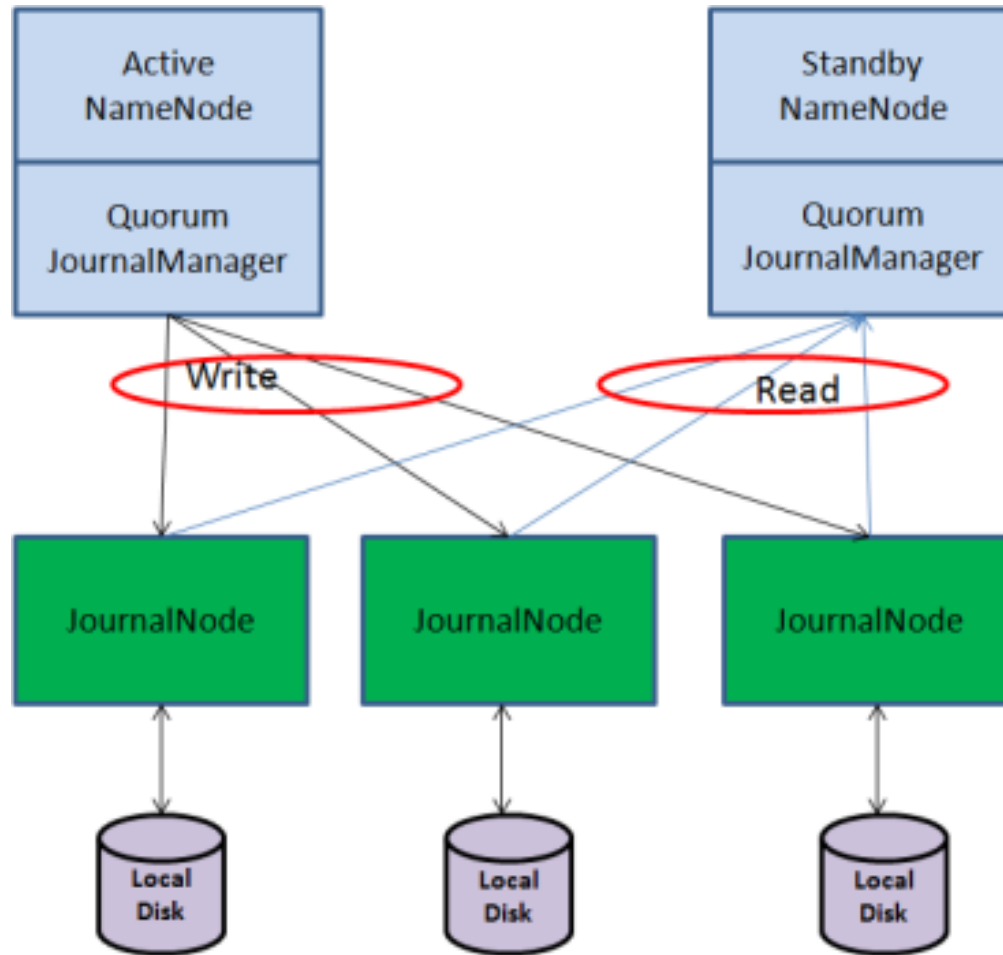
Metadata Disk Failure (Journal)

- Tipicamente, duas ou mais máquinas separadas são configuradas como NameNodes.
 - Uma sempre está com *status* **Active** e as demais em **Standby**.
 - Active NameNode é responsável por todas as operações do cliente no cluster.
 - Standbys mantem informações suficiente para fornecer um failover rápido, se necessário.
- **Standby** sincronizados com o **Active** através de um grupo de daemons separados denominados "JournalNodes" (JNs).
- Quando uma modificação do namespace é executada pelo nó **Active**, registra a modificação nos JNs.
- O **Standby** le as edições dos JNs e aplica as alterações em seu próprio Namespace.
- O **Standby** está constantemente observando as mudanças no **EditLog**.

Metadata Disk Failure (Journal) ^{FIAP}

- No caso de um **failover**, o **Standby** assegurará que tenha lido todas as edições dos **JournalNodes** antes de se promover ao estado **Active**, garantindo que o **namespace** seja totalmente sincronizado.
- O **Standby** deve ter informações atualizadas sobre a localização dos blocos no cluster.
 - **DataNodes** são configurados com a localização de todos os **NameNodes**
 - Envia informações de localização do bloco e **heartbeats** para todos **datanodes**.
- Apenas um dos **NameNode** está ativo por vez.
 - O **JournalNodes** só permite que um único **NameNode** seja escritor por vez.
 - Durante um failover, o **NameNode** que se tornará **Active** simplesmente assumirá o papel de escrever no **JournalNodes**, o que efetivamente evitará que o outro **NameNode** continue no estado **Active**, permitindo que o novo **Active** execute com segurança o **failover**.

Metadata Disk Failure (Journal)^{FIAP}



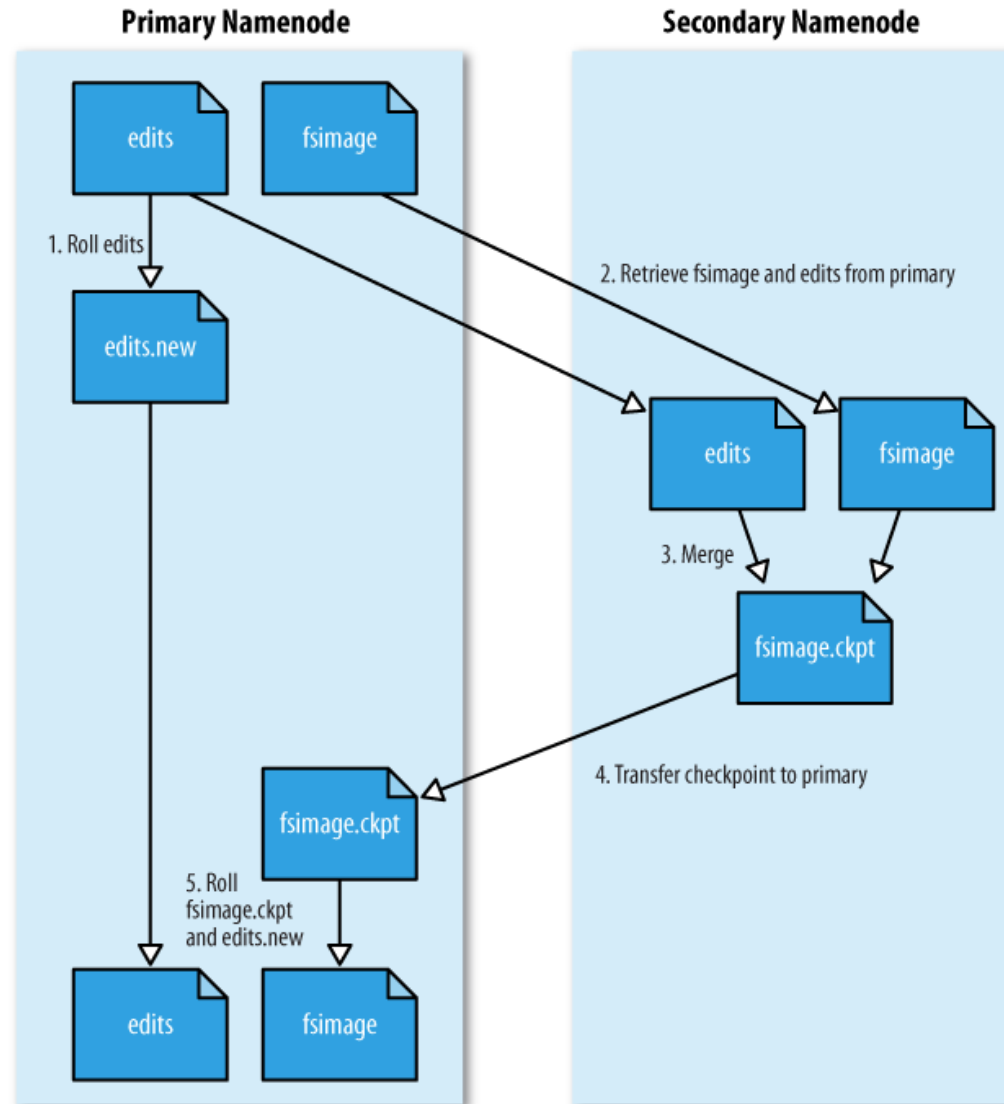
Secondary NameNode

- Não é um **hot standby** para o **NameNode**.
- Quando um **Namenode** é iniciado, ele lê os dados do arquivo **FsImage**, aplica as alterações a partir do arquivo **EditLogs** e, só então, grava o novo estado no arquivo **FsImage**.
 - No startup é executado uma operação de **merge** entre **Fsimage** e **Editlog**.
- As operações sempre são iniciadas com o arquivo **Editlog** vazio

Secondary NameNode

- Aplicar as informações do **Editlogs** pode levar algum tempo. **Secondary NameNode** resolve esse problema.
- Ele faz download dos arquivos **FsImage** e **EditLogs** do **NameNode**. E, em seguida, funde **EditLogs** com o **FsImage**, armazenando o **FsImage** em armazenamento persistente.
- Ele mantém o tamanho do **EditLogs** dentro de um limite.
- Executa **checkpoint** regular no HDFS.

Secondary NameNode



Secondary NameNode

```
cd /etc/hadoop/conf  
ls -ltr  
cat masters
```

- **masters** especifica o nome do **node** onde o **Secondary namenode** será iniciado.

```
hirwuser150430@ip-172-31-45-217:/etc/hadoop/conf$ cat masters  
ip-172-31-45-217.ec2.internal
```

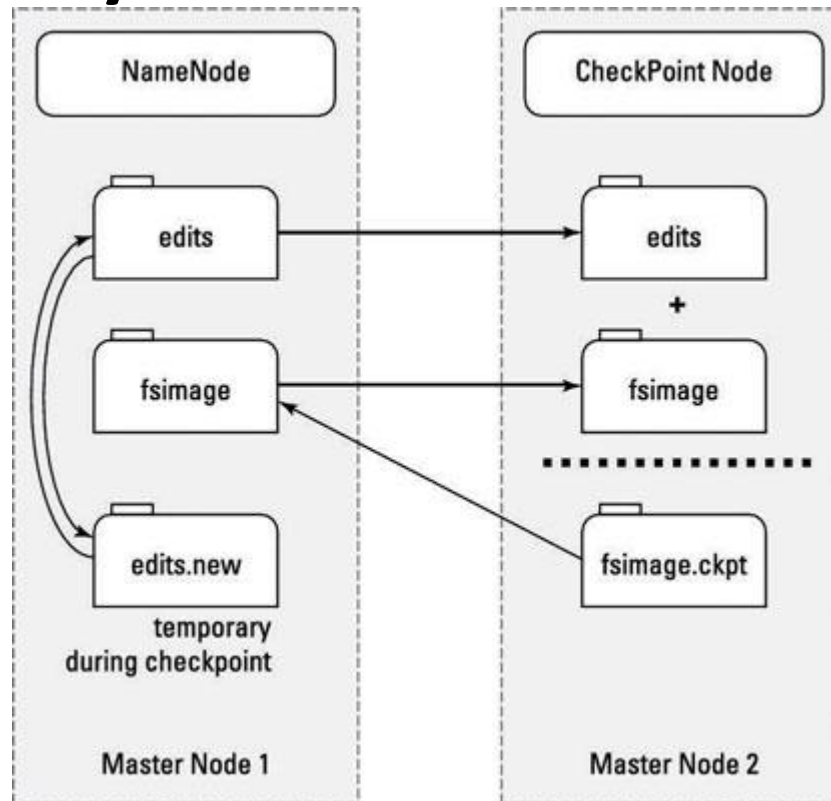
- **slaves** pode conter, opcionalmente, uma lista com todos os **datanodes**.

CheckPoint Node

- Periodicamente cria pontos de verificação do **namespace**.
- Faz download dos arquivos **FsImage** e **EditLog** do **Active Namenode**, ele combina-os localmente e ele carrega a nova imagem de volta para o **Active NameNode**.
- Armazena o **checkpoint** mais recente em um diretório que possui a mesma estrutura que o diretório do **Namenode**. Isso permite que a imagem verificada seja sempre disponível para leitura pelo **namenode**, se necessário.

CheckPoint Node

- A partir da versão 2.4, pode substituir o **Secondary NameNode**.

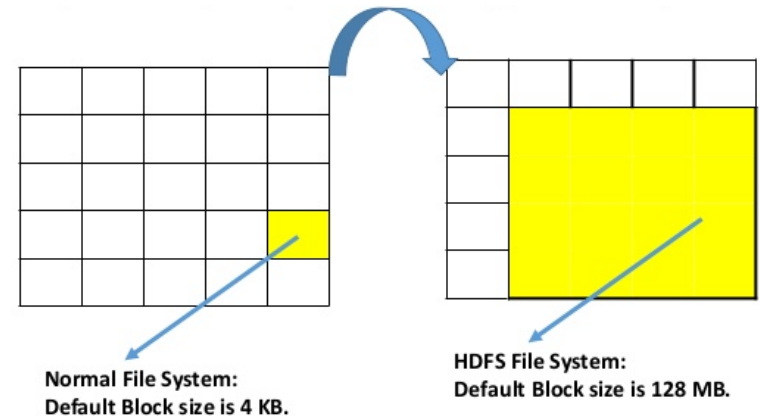


Backup Node

- Fornece a mesma funcionalidade de **checkpoint** que o **Checkpoint Node**, de maneira mais eficiente.
- Mantém uma cópia atualizada **in memory** do **namespace** do sistema de arquivos.
- Sempre sincronizado com o estado do **Active NameNode**.
- Não precisa baixar o **FsImage** e o **EditLog** do **Active NameNode** para criar um **checkpoint** pois já possui uma cópia memória.
- Mais eficiente, só precisa salvar o arquivo **FsImage** local e redefinir o **Editlog**.
- **NameNode** suporta apenas um **Backup Node**.

Blocos

- HDFS divide grandes arquivos pedaços menores conhecidos como Blocos.
 - menor unidade de dados em um sistema de arquivos.
 - Clientes e administradores não têm controle sobre o bloqueio e localização do bloco.
 - NameNode decide todas essas coisas.
- O tamanho padrão de 128 MB, que pode ser configurado conforme a necessidade.
- Todos os blocos do arquivo são do mesmo tamanho, exceto o último bloco, que pode ser do mesmo tamanho ou menor.



Replicação

- Replicação de blocos fornece tolerância a falhas.
- Se uma cópia não for acessível ou estiver corrompida, pode ser lida de outra cópia.
- O número de cópias ou réplicas de cada bloco de um arquivo é o fator de replicação (3, por padrão). Cada bloco é replicado três vezes e é armazenado em diferentes **DataNodes**.
- Em uma configuração padrão, um arquivo de 128 MB em HDFS ocupa 384 MB ($3 * 128 \text{ MB}$) de espaço.
- **NameNode** recebe relatório de bloco do **DataNode** periodicamente para manter o fator de replicação.
- Se um bloco estiver super-replicado / sub-replicado, o **NameNode** adiciona ou exclui as réplicas conforme necessário.

Replicação

– `hadoop fs -ls teste3-seunome`

```
hirwuser150430@ip-172-31-45-217:~$ hadoop fs -ls teste3-seunome
Found 1 items
-rw-r--r--  3 hirwuser150430 hirwuser150430    3326129 2017-10-17 01:46 teste3-seunome/dwp-payments-april10.csv
```

– `hadoop fs -Ddfs.replication=2 -cp
teste2-seunome/dwp-payments-
april10.csv teste2-
seunome/test_with_rep2.csv`

– `hadoop fs -ls teste2-seunome`

– `hadoop fs -ls teste2-
seunome/test_with_rep2.csv`

```
hirwuser150430@ip-172-31-45-217:~$ hadoop fs -ls teste2-seunome
Found 2 items
-rw-r--r--  3 hirwuser150430 hirwuser150430    3326129 2017-10-17 01:49 teste2-seunome/dwp-payments-april10.csv
-rw-r--r--  2 hirwuser150430 hirwuser150430    3326129 2017-10-17 01:51 teste2-seunome/test_with_rep2.csv
```

Visualizando o Node

<http://ec2-54-92-244-237.compute-1.amazonaws.com:50070/dfshealth.html#tab-overview>

Hadoop	Overview	Datanodes	Datanode Volume Failures	Snapshot	Startup Progress	Utilities ▾
--------	----------	-----------	--------------------------	----------	------------------	-------------

Overview 'ip-172-31-45-216.ec2.internal:8020' (active)

Started:	Tue Aug 15 09:08:58 -0300 2017
Version:	2.6.0-cdh5.11.0, r91a488f2c5abb3de0e6ee74080dbc439c7576fb4
Compiled:	Thu Apr 06 00:07:00 -0300 2017 by jenkins from Unknown
Cluster ID:	CID-9fc14d1e-1227-4584-bcd0-b425322e9cb7
Block Pool ID:	BP-2125152513-172.31.45.216-1410037307133

Exercício

- Liste o diretório ROOT do HDFS
- Liste o diretório HOME default
- Crie um diretório no HDFS com o nome de “SeuNome1”. Onde “SeuNome” é o seu nome.
- Copie um arquivo qualquer do sistema local para o diretório HDFS que acabou de criar.
- Crie mais dois diretórios com nome de “SeuNome2” e “SeuNome3”.
- Copie o arquivo de “SeuNome1” para “SeuNome2”.
- Mova o arquivo de “SeuNome1” para “SeuNome3”.
- Apague os diretórios e arquivos que criou do HDFS.