HADOOP

INTRODUZIONE

Infrastrutture Big data

How *big* are Big Data?

Il volume globale di dati nel 2012 è stato:

- 3 Zettabytes …
- 3,000 Exabytes ...
- 3,000,000 Petabytes ...
- 3,000,000,000 Terabytes …
- 3,000,000,000,000,000,000,000 bytes!!

Google processa più di 100 PB al giorno con 3M di server

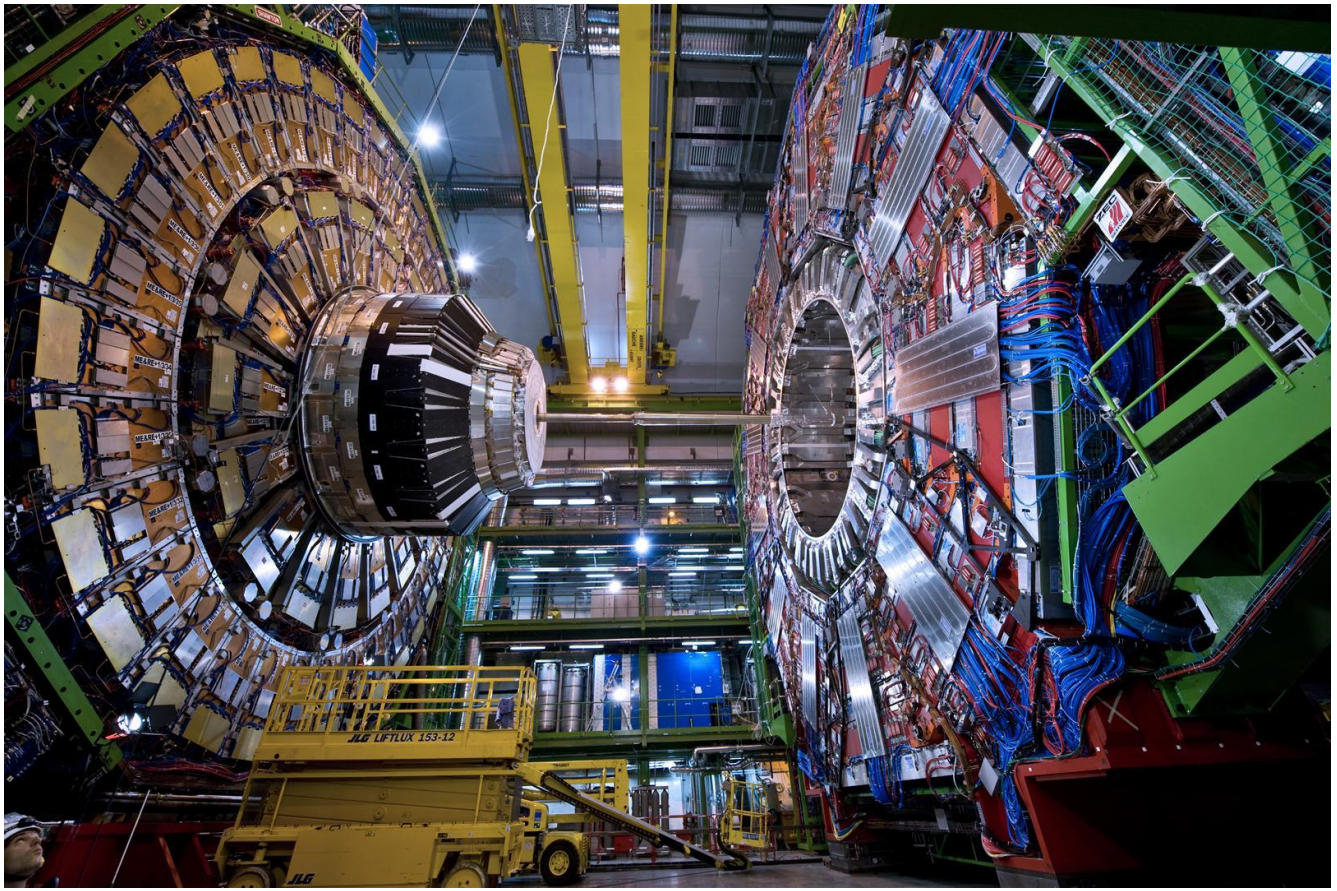
Facebook possiede 300 PB di user data + 500 TB/giorno

YouTube 1000 PB archivio video

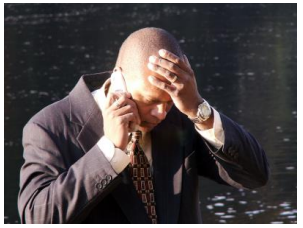
1.4B US carte di credito, 20B transazioni/anno

CERN LHC genera 15 PB all'anno

Come processiamo i Big Data?



Cattive notizie



Capacità di archiviazione sono cresciute negli anni, ma la velocità di accesso al disco no

Year: 1990
Size: 1.3 GB
Speed: 4,4 MB/s

5 minutes

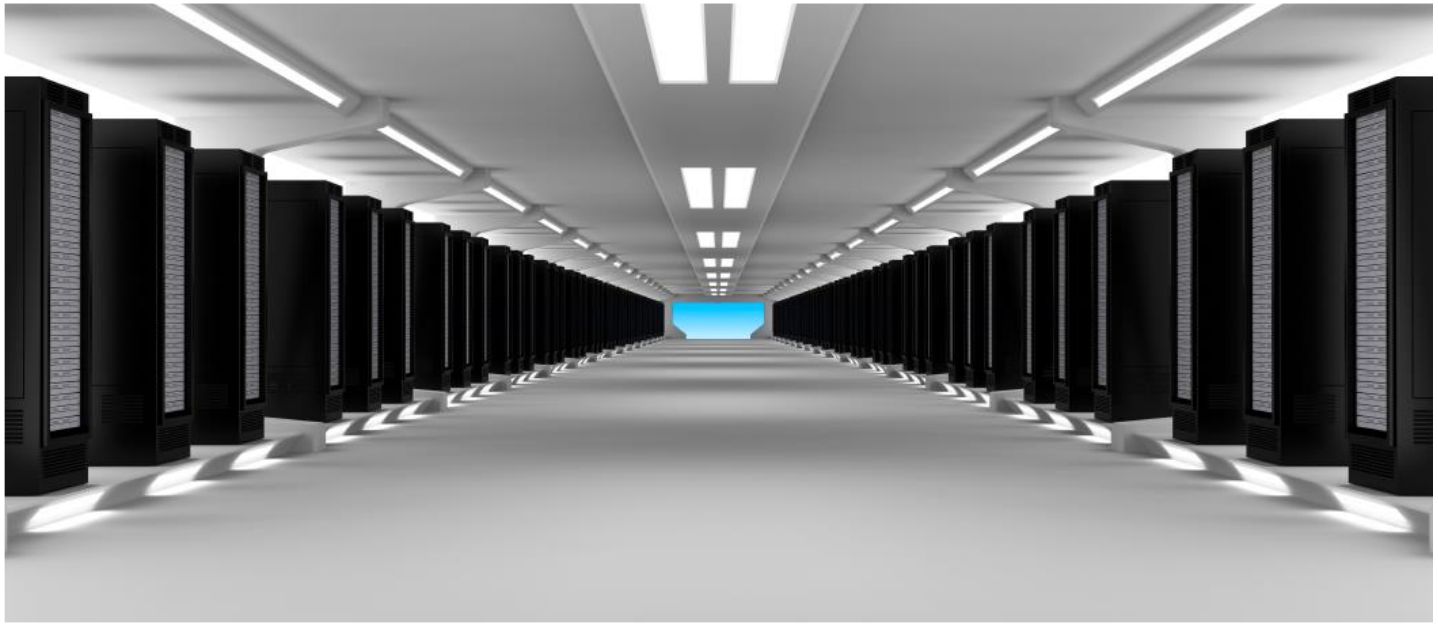
Year: 2014
Size: 1 TB
Speed: 100 MB/s

3 hours

Year: 2015
Size: 1 TB
Speed: 600 MB/s

30 minutes

Soluzione:
cluster computing



100 hard disk? 2 min per leggere 1TB

Scale up

Si riferisce all'aggiunta di CPU e RAM, necessità di comprare un server più robusto

Pro

- Meno consumo energetico rispetto ad aggiungere molti server
- Costi di raffreddamento più bassi
- Meno difficili da installare e mantenere
- Minori costi di licenze
- Minore strumentazione di rete necessaria

Contro

- PREZZO!
- In caso di failure c'è maggiore rischio di interruzione di servizio
- Sensibili a lock-in
- Ci sono limitazioni negli aggiornamenti futuri

Scale out

Si riferisce all'aggiunta di più server, ciascuno con CPU e RAM ridotte

- Di solito più economico
- Può letteralmente scalare all'infinito

Pro

- Più economico di scale up
- Le nuove tecnologie semplificano la tolleranza ai guasti e il monitoraggio
- Facile da aggiornare

Contro

- Maggiori costi di licenza
- Maggiore ingombro nel Data Center
- Costi di utenza più elevati (elettricità e raffreddamento)
- Possibile necessità di più apparecchiature di rete (switch/router)

Cluster computing

I nodi di calcolo vengono archiviati nei rack

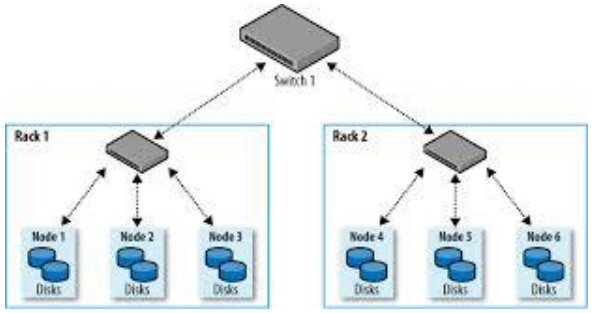
8 – 64 nodi di calcolo su un rack

Possono esserci molti rack di nodi di calcolo

- I nodi di un singolo rack collegati da rete veloce
- I rack sono collegati da un altro livello di rete (switch)
- La larghezza di banda della comunicazione intra-rack è di solito molto superiore a quella della comunicazione inter-rack

I nodi di calcolo possono fallire! Soluzione:

- Computazione suddivisa in task
- File archiviati in modo ridondante



Commodity hardware

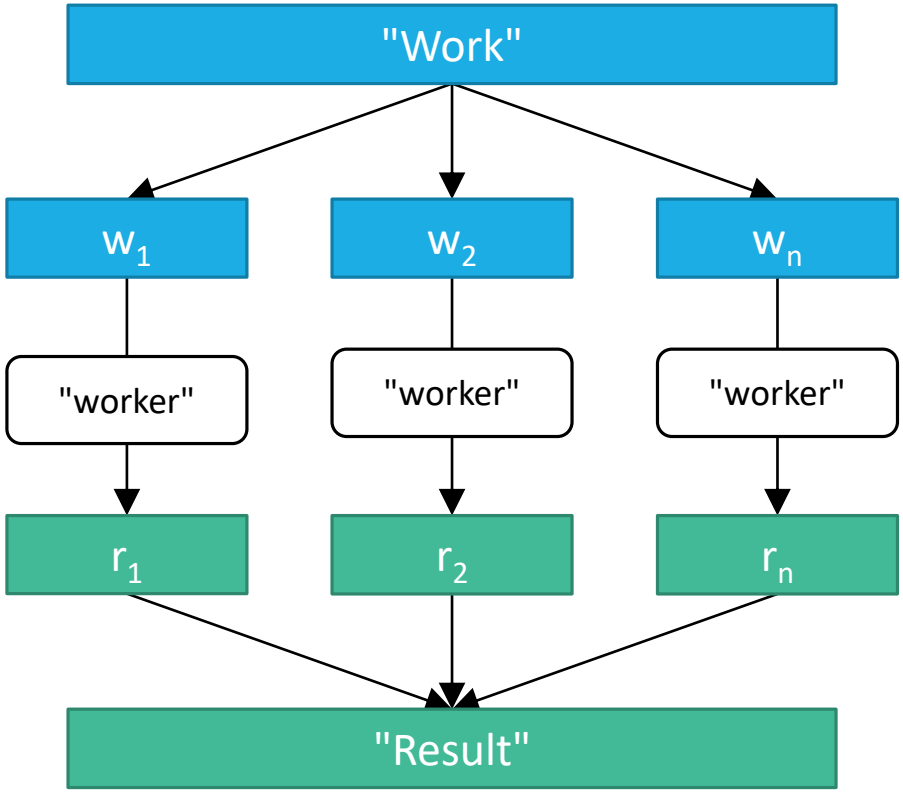
Esempio di specifiche tipiche di hardware per materie prime:

- Processore: 2 quad-core 2-2.5GHz CPUs
- Memoria: 16-24 GB ECC RAM
- Archiviazione: 4 × 1TB SATA disks
- Rete: Gigabit Ethernet

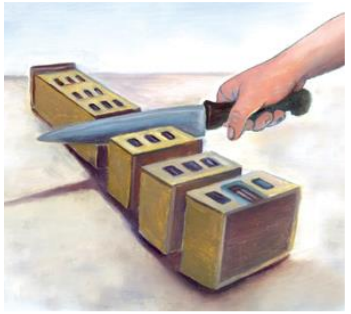
Yahoo! ha un'enorme installazione di Hadoop:
> 100,000 CPU in > 60,000 computer (2017)

- Utilizzato per supportare sistemi pubblicitari e ricerca Web
- Utilizzato per supportare scaling test per sviluppo di Hadoop

Calcolo distribuito: una vecchia idea



Divide



Conquer

Sfide di parallelizzazione

1. Come si assegnano le unità di lavoro ai lavoratori?
2. Cosa succede se abbiamo più unità di lavoro che lavoratori?
3. Cosa succede se i lavoratori devono condividere risultati parziali?
4. Come aggreghiamo i risultati parziali?
5. Come facciamo a sapere che tutti i lavoratori hanno finiti?
6. E se i lavoratori muoiono?

Qual è il rischio di tutti questi problemi?

Sfide di parallelizzazione

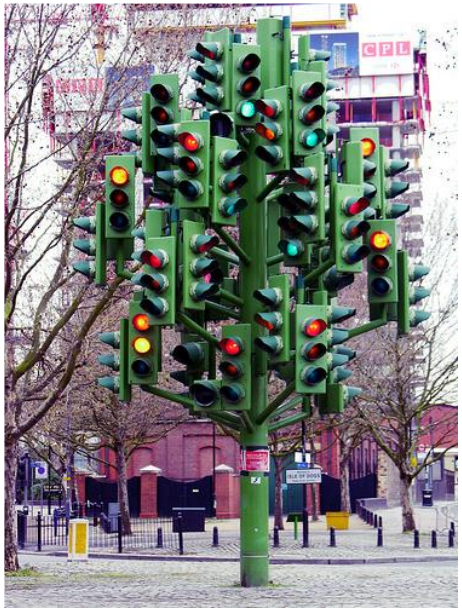
Rischi?

Molto: per esempio, deadlock e starvation

I problemi di parallelizzazione derivano da:

- Comunicazione tra lavoratori (ad esempio, per lo scambio di stato)
- Accesso alle risorse condivise (ad es. dati)

Abbiamo bisogno di un
meccanismo di sincronizzazione



Qual è il punto?

Nascondi i dettagli a livello di sistema dallo sviluppatore

- Non più race conditions, lock contention, etc.

Separare “cosa” da “come”

- Lo sviluppatore specifica il calcolo che deve essere eseguito
- Il Framework di esecuzione ("Runtime") gestisce l'esecuzione effettiva

Il datacenter è il computer!

Architetture di elaborazione parallela

Calcolo parallelo

L'elaborazione parallela può essere ottenuta attraverso diverse architetture:

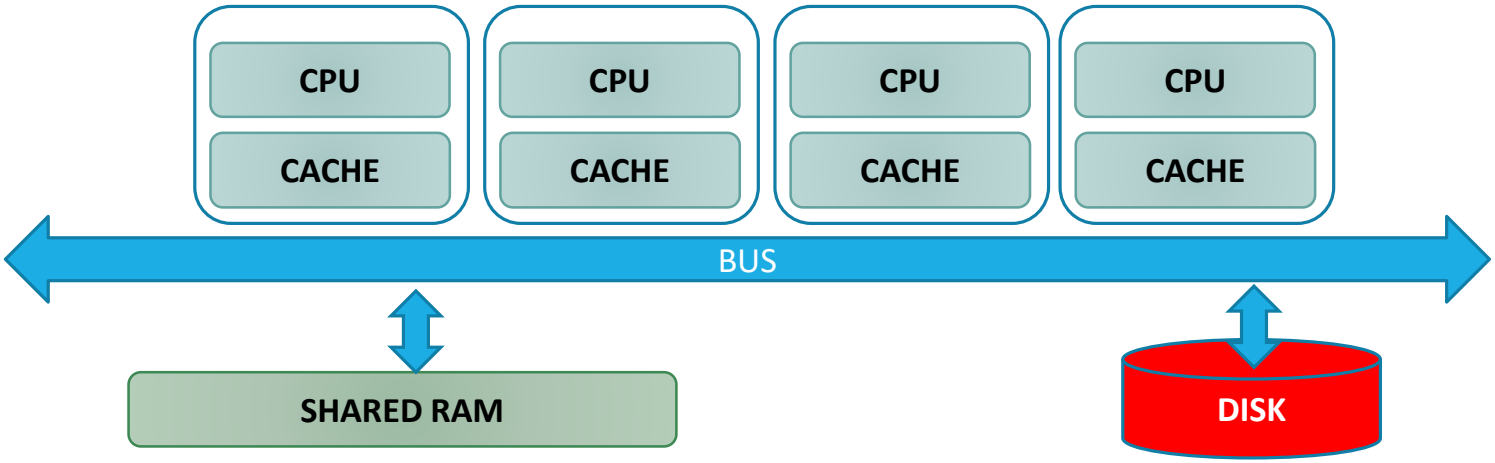
- Symmetric Multiprocessors (SMP)
- Massively Parallel Processors (MPP)
- **Clusters**
- Grids
- ... e altri
  - Non-Uniform Memory Access (NUMA)
  - Distributed Shared Memory (DSM)
  - Network of Workstation (NoW)
  - Constellations
  - ...

SMP Architecture

Symmetric Multi Processing (SMP) è l'architettura adottata da RDBMS tradizionale: diversi processori condividono la stessa RAM, lo stesso bus I/O e lo stesso disco (s)

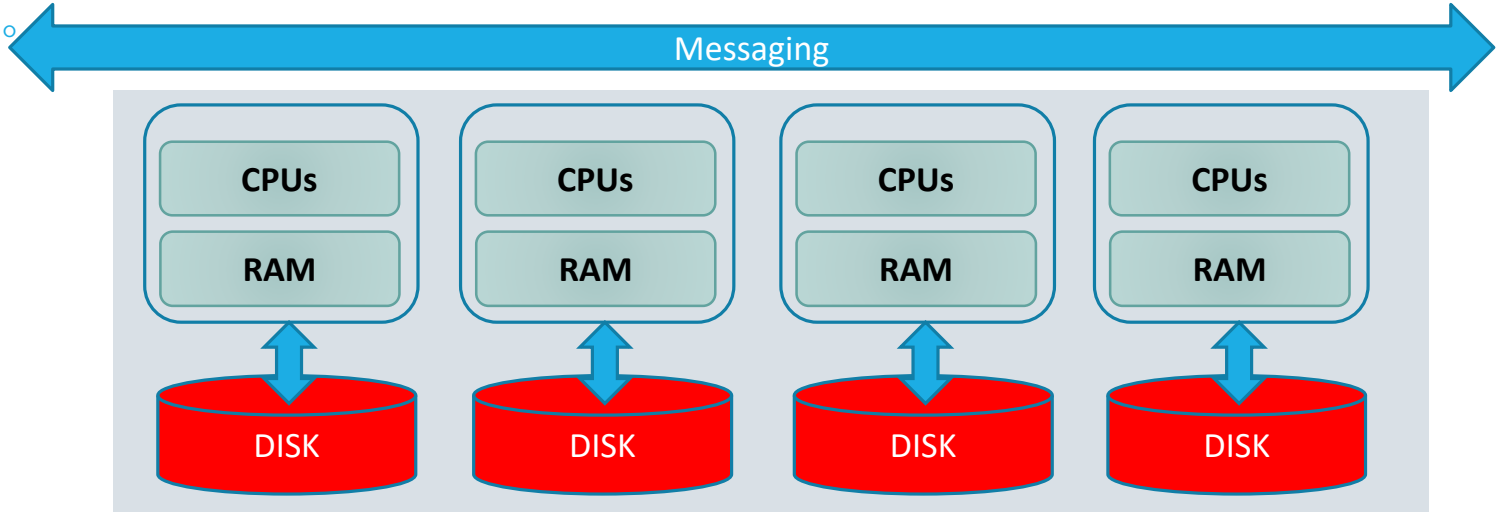
Limiti

- Numero fisico di dispositivi che possono essere montati
- Collo di bottiglia del BUS



MPP Architecture

In una *Massively Parallel Processing* (MPP) architettura ci sono diversi processori, dotati di RAM e dischi propri, collaborando per risolvere un singolo problema suddividendolo in task indipendenti

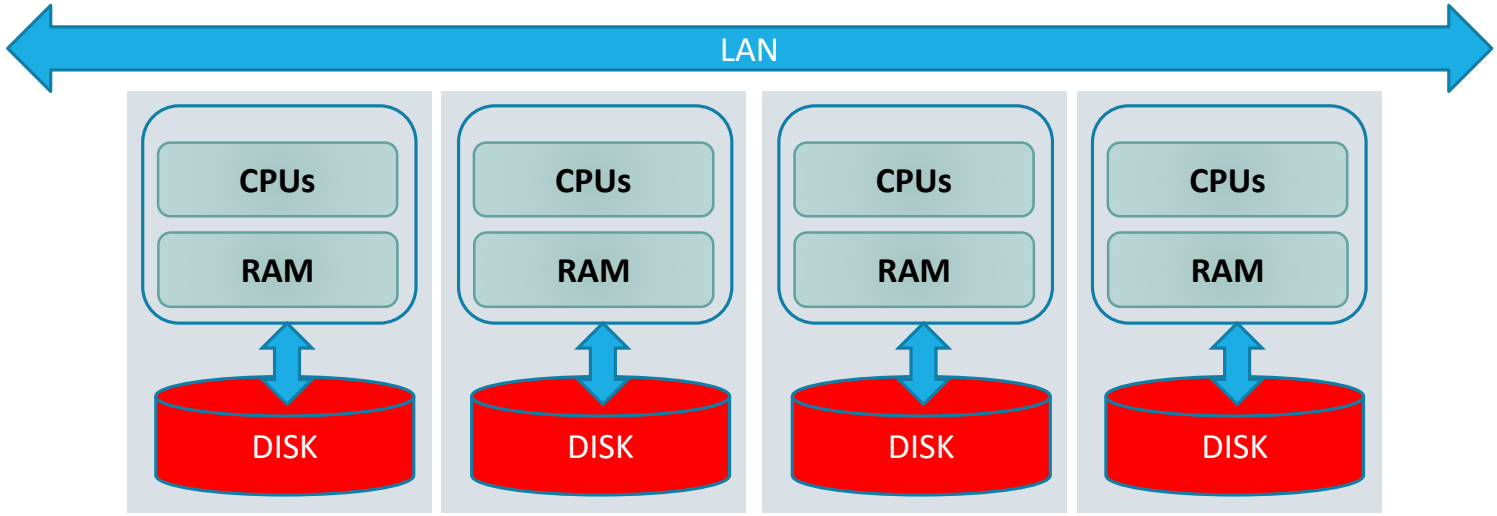
- Shared nothing architecture
- Utilizzato principalmente per applicazioni di data warehouse (ad es., Teradata, Nettezza, Vertica)
- 

Architettura di cluster computing

Un cluster di computer è un gruppo di computer collegati (nodi), che collaborano formando un singolo computer

Tipicamente collegati tra loro tramite LAN veloci

- Ogni nodo è un sistema a sé, capace di operazioni indipendenti
- Numero di nodi nel cluster > > numero di CPU in un nodo



Grid Computing Architettura

Il Grid Computing è la raccolta di risorse informatiche da più sedi per raggiungere un obiettivo comune

Il Grid computing si distingue dal cluster computing in:

- Ogni nodo è impostato per eseguire un task/applicazione diversa
- I sistemi di grid computing tendono ad essere più eterogenei e geograficamente dispersi (quindi non accoppiati fisicamente) rispetto ai computer cluster
- Anche se una singola griglia può essere dedicata a una particolare applicazione, è comunemente usata per scopi diversi
- Le griglie sono spesso costruite con librerie software middleware di griglia di uso generico

Apache Hadoop

Che cos'è Hadoop?

Un Framework software open source (Apache project)

- Originariamente sviluppato da Yahoo!, ispirato da Google
- Goal: archiviazione ed elaborazione di set di dati su vasta scala

Infrastruttura: cluster di hardware per materie prime

Moduli:

- **Common**, un insieme di librerie e Utility (servizi essenziali, Jar necessari, ecc.)
- **HDFS**, un file System distribuito
- **MapReduce**, modello di programmazione per l'elaborazione big data
- **YARN**, una piattaforma di gestione delle risorse

Include una serie di progetti

- Apache Pig, Apache Hive, Apache HBase, etc..

Utilizzato in produzione da Google, Facebook, Yahoo! e molti altri

Una definizione

"The Apache Hadoop software library is a framework that allows for the **distributed processing** of **large data sets** across **clusters of computers** using simple programming models."

È progettato per scalare da singoli server a migliaia di macchine, ognuna delle quali offre calcoli e storage locali.

Piuttosto che affidarsi all'hardware per fornire **disponibilità elevata** e **affidabilità**, il framework è progettato per **rilevare e gestire gli errori a livello di applicazione**, fornendo un servizio altamente disponibile in cima a un cluster di computer (ognuno dei quali può essere soggetto a guasti).

http://hadoop.apache.org/

Hadoop architecture (1)

Common

- Le utilità comuni che supportano gli altri moduli Hadoop

HDFS (Hadoop Distributed File System)

- Un file System distribuito che fornisce un accesso ad alta velocità ai dati delle applicazioni
- I dati vengono replicati con ridondanza in tutto il cluster

Hadoop architecture (2)

YARN (Yet Another Resource Negotiator)

- Framework per la pianificazione dei processi e gestione risorse cluster

Map Reduce:

- Un sistema basato su YARN per l'elaborazione parallela di big data
- Motore della maggior parte dei Big Data Processing
- Utilizzato anche in altri ambienti MPP e database NoSQL (ad esempio, Vertica e MongoDB)

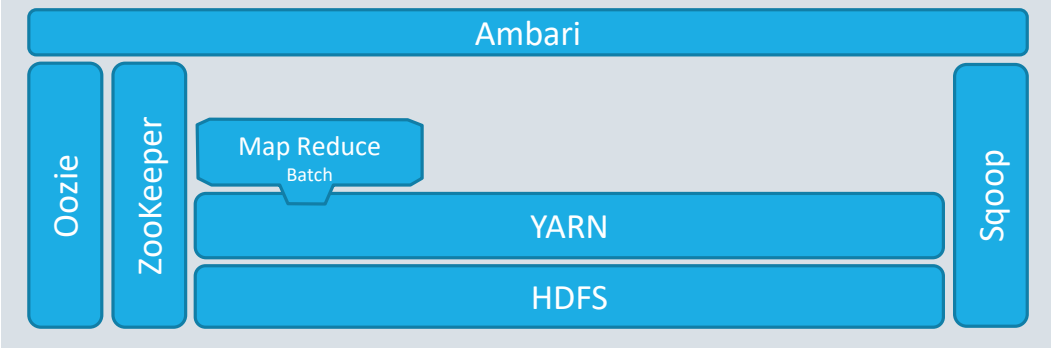
Hadoop architecture (3)

Apache Ambari: strumento basato sul Web per il provisioning, la gestione e il monitoraggio dei cluster Hadoop; include il supporto per molte applicazioni

Apache ZooKeeper: servizio di coordinamento ad alte prestazioni per applicazioni distribuite

Apache Oozie: sistema di pianificazione del flusso di lavoro per gestire i processi Hadoop

Apache Sqoop: utilizzato per trasferire in modo efficiente i dati bulk tra Hadoop e le origini dati strutturate (ad esempio, RDBMSs)



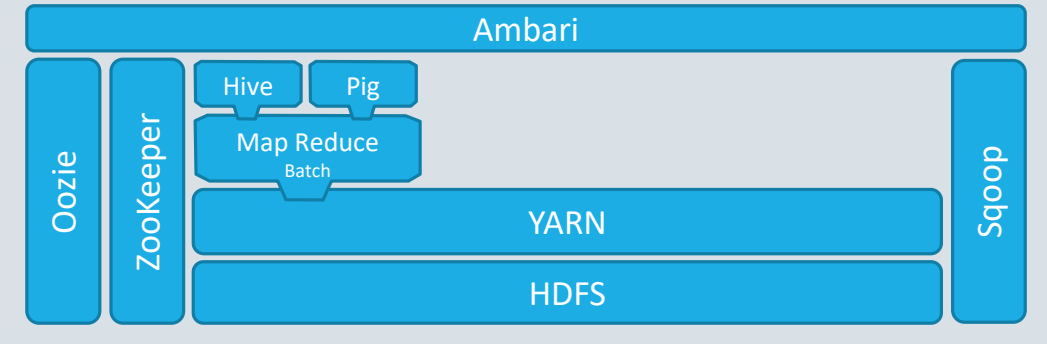
Hadoop architecture (4)

Apache Pig: una piattaforma per l'analisi di grandi set di dati

- I programmi assomigliano a flussi ETL e vengono compilati in processi MapReduce

Apache Hive: data warehouse software

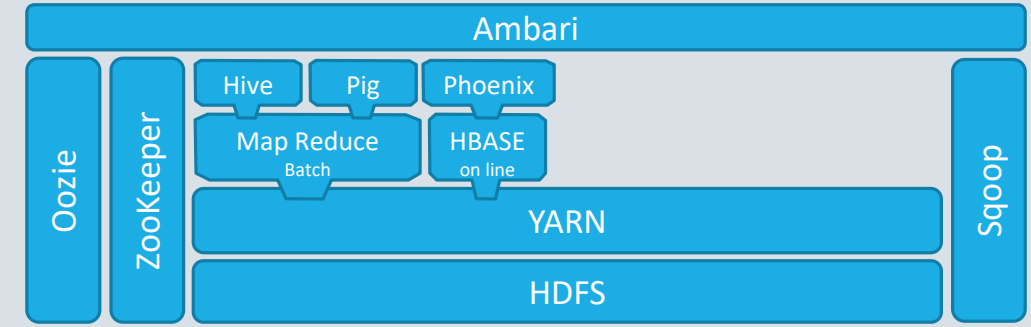
- Struttura dei progetti su dati HDFS archiviati in diversi formati
- Le query vengono espresse in HiveQL (SQL-like) e compilate in processi MapReduce
- Preferito dagli analisti dei dati



Hadoop architecture (5)

Apache HBase: DBMS non relazionale e distribuito

Phoenix: un livello di database relazionale su HBase



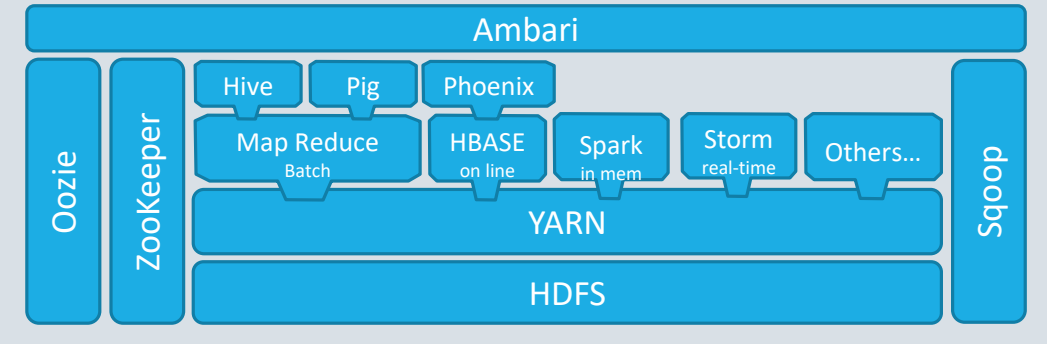
Hadoop architecture (6)

Apache Spark: MapReduce-like engine

- Le primitive in memoria di Spark offrono prestazioni fino a 100 volte più veloci
- Adatto agli algoritmi di apprendimento automatico, supporta query veloci
- Gestisce anche lo streaming dei dati, l'elaborazione dei grafici e l'astrazione SQL

Apache Storm: calcolo in tempo reale per elaborare i dati di streaming

- Potente per analisi in tempo reale, Machine Learning, monitoraggio continuo



Hadoop architecture (7)

Apache Giraph: graph processing

- Attualmente utilizzato su Facebook per analizzare il grafico sociale formato dagli utenti e le loro connessioni.

Apache Mahout: algoritmi di machine learning

Apache Kafka: piattaforma streaming

Cloudera Impala: MPP SQL query engine

HDFS

HADOOP DISTRIBUTED FILE SYSTEM

HDFS

HDFS è un filesystem progettato per archiviare file di grandi dimensioni, in esecuzione su cluster di commodity hardware

- Un file tipico in HDFS è Gigabyte a Terabyte di dimensioni. Ci sono cluster Hadoop in esecuzione oggi che memorizzano petabyte di dati
- Le applicazioni eseguite su HDFS necessitano di accesso in streaming ai loro set di dati. HDFS è progettato più per l'elaborazione in batch piuttosto che l'uso interattivo. L'enfasi è sull'elevata produttività dell'accesso ai dati piuttosto che sulla bassa latenza di accesso ai dati
- Le applicazioni HDFS necessitano di un modello di accesso Write-Once-Read-Many per i file. Un file una volta creato, scritto e chiuso non deve essere modificato. Questo presupposto semplifica i problemi di coerenza dei dati e consente l'accesso ai dati a throughput elevato
- L'errore hardware è la norma piuttosto che l'eccezione. Pertanto, il rilevamento di guasti e il rapido recupero automatico è obiettivo di base di HDFS

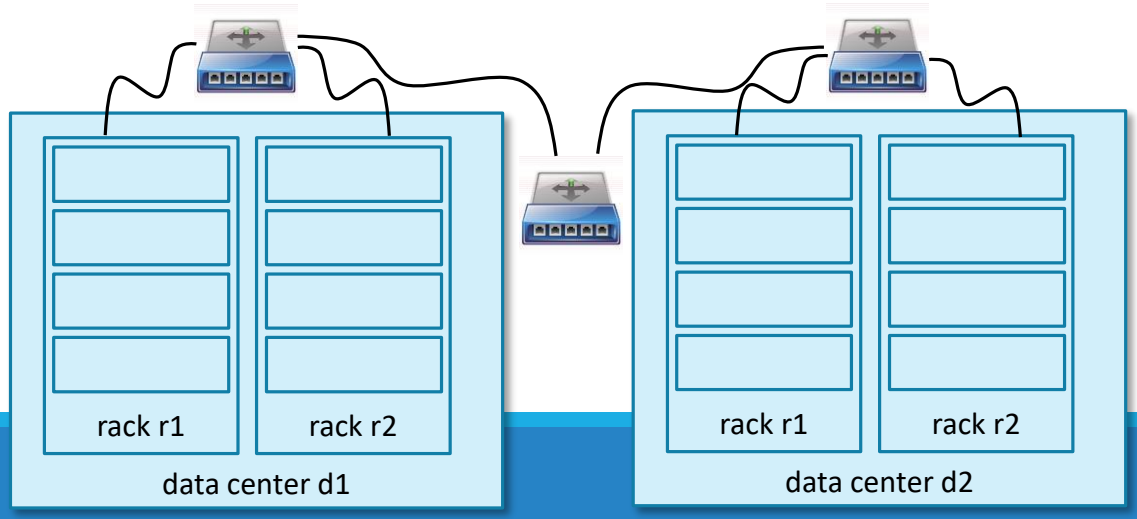
HDFS: Topologia cluster

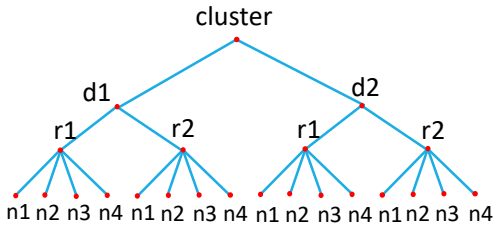
Hadoop deve essere consapevole della topologia del cluster definita durante la fase di impostazione del cluster.

- Storage a blocchi e allocazione dei processi (località dati) necessitano di tali informazioni

I **nodi** sono organizzati in **rack**, i rack sono organizzati in **data center**

Hadoop modella tali concetti in modo simile ad un albero e calcola la distanza tra i nodi come la loro distanza sull'albero.





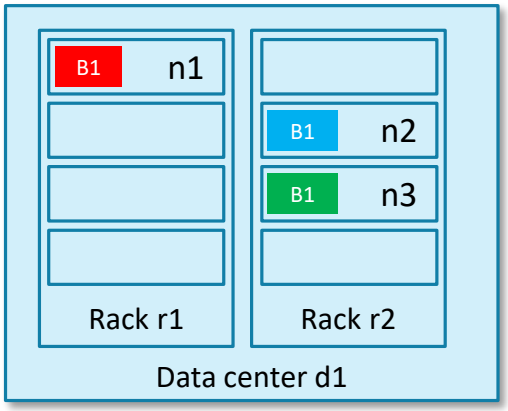
HDFS: Replica

Ogni blocco di dati viene replicato in modo indipendente in più DNs al fine di migliorare le prestazioni e robustezza

- La replica è consapevole della topologia del cluster
- Per ogni blocco dati, il NN memorizza l'elenco dei DNs

Il fattore di replica predefinito è 3:

- Replica 1 è memorizzato sul nodo (N1) dove il client ha emesso il comando di scrittura (se il client risiede all'interno del cluster)
- Replica 2 è memorizzato su un nodo (N2) in un rack (R2) diverso da quello di N1 (off-rack)
- Replica 3 è memorizzato su un nodo, diverso da N2, ma che appartiene a R2



- Le repliche possono essere riequilibrate quando i nodi vengono aggiunti o non disponibili

Data Locality

Hadoop sfrutta la topologia del cluster e la replica del blocco dati per applicare il principio della località dati

«*When computations involves large set of data,*
it is cheaper (i.e. faster) to move code to data
rather than data to code”

I seguenti casi rispettano l'ordine che il gestore delle risorse preferisce:

1. Elaborazione e dati sullo stesso nodo
2. Elaborazione e dati sui diversi nodi dello stesso rack
3. Elaborazione e dati su diversi rack dello stesso Data Center
4. Elaborazione e dati su diversi rack dei diversi Data Center

HDFS: non sempre la scelta migliore

Anche se questo può cambiare in futuro, ci sono applicazioni in cui HDFS non è consigliato

Accesso ai dati a bassa latenza

- HDFS è ottimizzato per fornire un throughput elevato di dati, e questo può essere a scapito della latenza. Le applicazioni che richiedono l'accesso ai dati nell'intervallo di decine di millisecondi non funzioneranno bene con HDFS.

Tanti di file di piccole dimensioni

- Il limite al numero di file in un filesystem è regolato dalla quantità di memoria sul NN, perché contiene i metadati del filesystem in memoria.
- Anche se la memorizzazione di milioni di file è fattibile, la gestione di miliardi è oltre la capacità corrente.

YARN

YET ANOTHER RESOURCE NEGOTIATOR

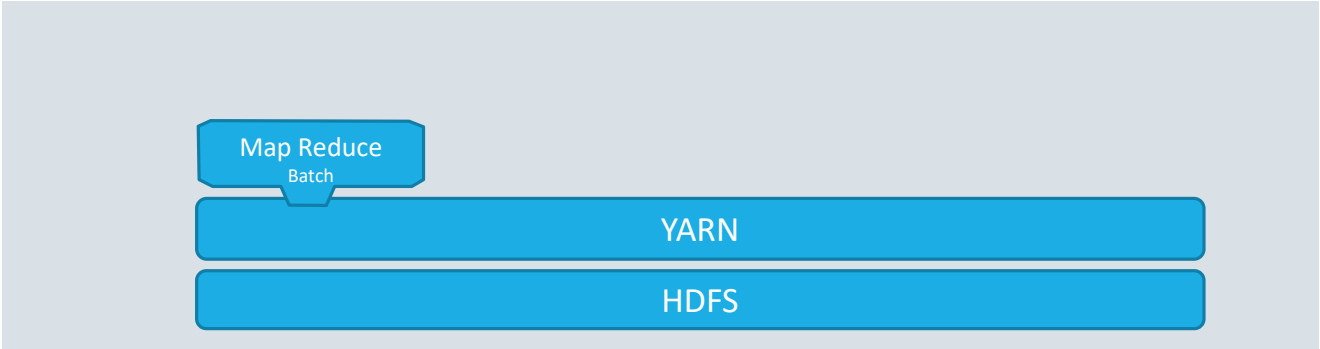
YARN

Apache YARN è la gestione delle risorse del cluster di Hadoop

- Introdotto in Hadoop 2 per migliorare l'implementazione di MapReduce, ma è abbastanza generale per supportare altri paradigmi di calcolo distribuito
- In Hadoop 1, le funzionalità di pianificazione/monitoraggio del lavoro sono state entrambe prese dal Job Tracker

YARN fornisce le API per richiedere e lavorare con le risorse del cluster

- Queste API vengono in genere utilizzate dai framework di elaborazione distribuita (ad esempio MapReduce, Spark), non dal codice utente



YARN: demoni

YARN fornisce i suoi servizi principali tramite due tipi di demoni:

Globale (uno per cluster), **Resource Manager** (RM)

- È l'autorità suprema che arbitra le risorse tra tutte le applicazioni

Uno per slave, **Node Manager** (NM)

- È responsabile dei container; monitora il loro utilizzo delle risorse (CPU, memoria, disco, rete, ecc.) e li riporta al RM
- Un **container** è un'entità astratta, utilizzata per eseguire un processo specifico dell'applicazione con un insieme vincolato di risorse

Su richiesta di un cliente, RM trova un NM che può lanciare *application master* process (AMP) in un container

- L'AMP può quindi richiedere più contenitori per eseguire un calcolo distribuito; la richiesta di allocazione/deallocazione può avvenire in modo dinamico

YARN: Resource Manager

L'RM ha due componenti principali:

Scheduler responsabile dell'allocazione delle risorse alle varie applicazioni in esecuzione, in base ai requisiti delle risorse.
Si tratta di uno scheduler puro, nel senso che:

- Non esegue alcun monitoraggio o monitoraggio dello stato per l'applicazione
- Non offre alcuna garanzia sul riavvio delle attività non riuscite a causa di errori dell'applicazione o hardware

Application Manager è responsabile dell'accettazione delle candidature, negoziando il primo container per l'esecuzione dell'AMP e fornendo il servizio per il riavvio dell'AMP in caso di guasto.

YARN: Scheduler

YARN offre una scelta di pianificatori e criteri configurabili

FIFO Scheduler

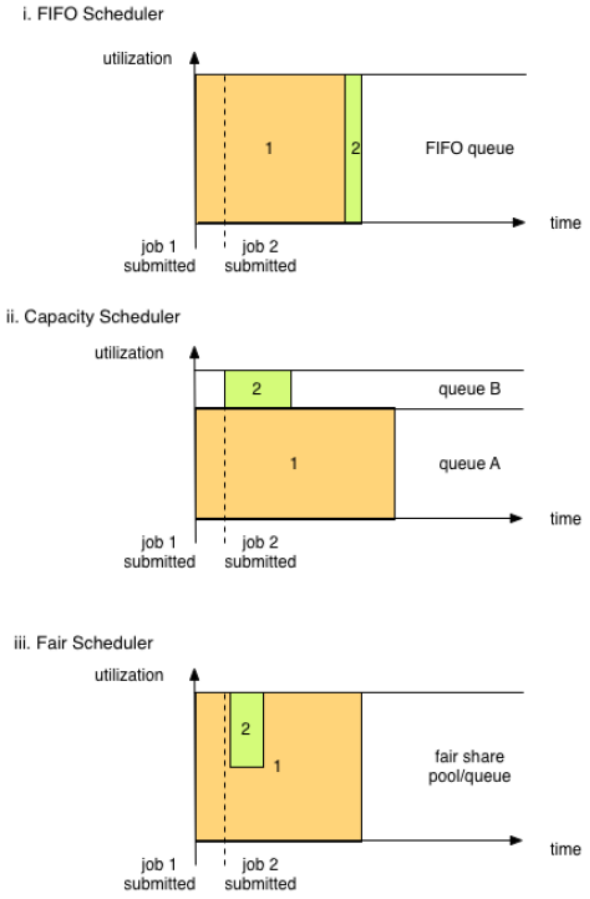
- Semplice da capire, nessuna configurazione necessaria
- Non adatto per cluster condivisi

Capacity Scheduler (default)

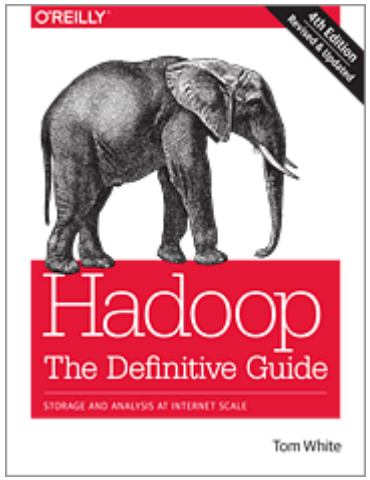
- Riserva una quantità fissa di capacità a ciascun job

Fair Scheduler (default in CDH)

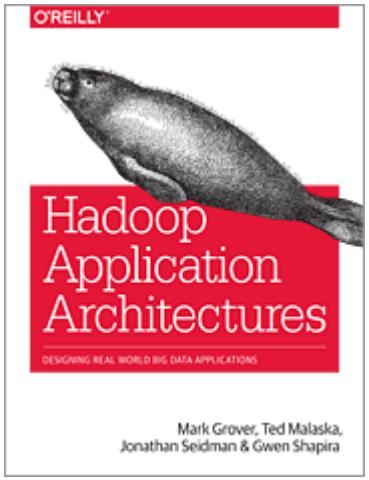
- Bilancia dinamicamente le risorse disponibili tra tutti i lavori in esecuzione



Suggested reading and resources



T. White
Hadoop The Definitive Guide
4th edition (O'Reilly)



G. Shapira, J. Seidman,
T. Malaska, M. Grover
Hadoop Application Architectures
(O'Reilly)

Hadoop official documentation
https://hadoop.apache.org/docs/stable/hadoop-project-dist