



Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

BIG DATA e Analisi dei Dati

Lezione 3.1 – Hadoop MapReduce

Mauro Bellone,
Robotics and AI researcher

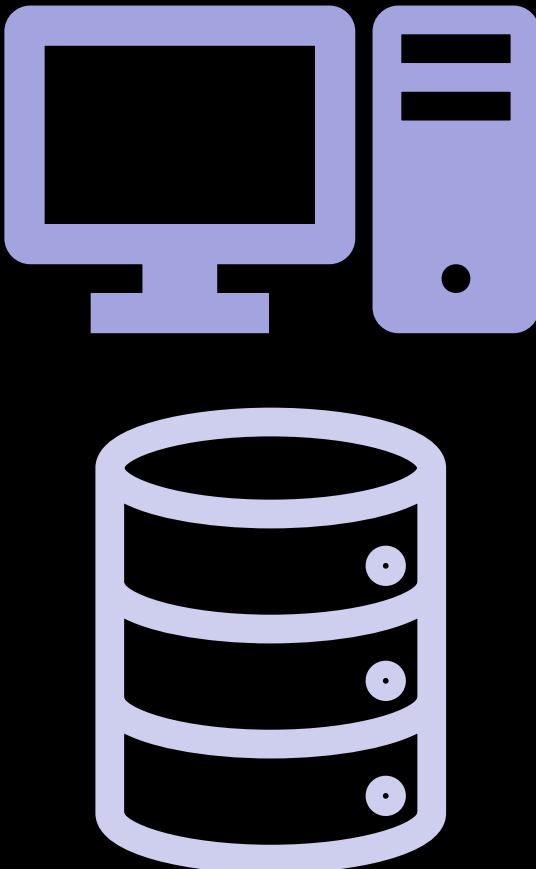
bellonemauro@gmail.com
www.maurobellone.com

Obiettivo

- ✓ Comprensione dell'architettura di mapReduce
- ✓ Introduzione ai componenti principali di mapReduce

Hadoop mapReduce

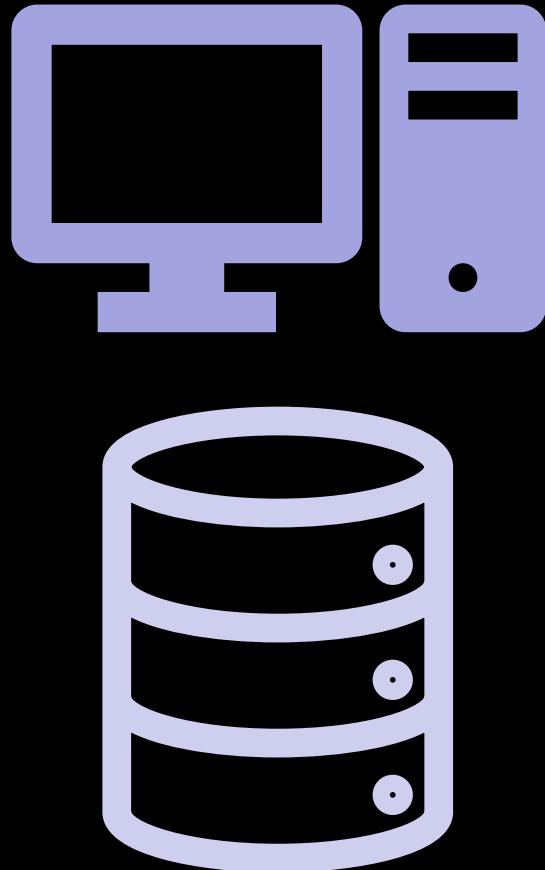
Grandi quantità di dati in un singolo server
semplicemente non ci entrano!!!



La possibilità di avere perdite di dati e disporre di
backup continui riduce la scalabilità dei singoli server

Hadoop mapReduce

Grandi quantità di dati in un singolo server
semplicemente non ci entrano!!!



La possibilità di avere perdite di dati e disporre di backup continui riduce la scalabilità dei singoli server

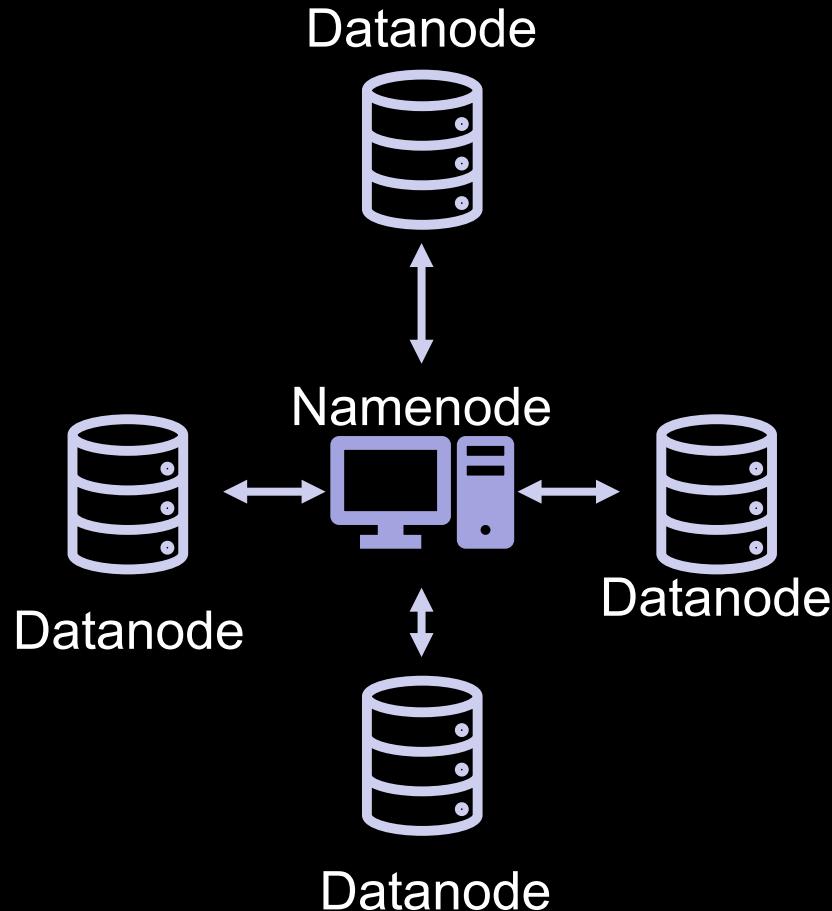


Con mapReduce le queries possono essere eseguite simultaneamente su molti server e i risultati possono essere logicamente integrati, permettendo l'analisi dei dati in tempo reale

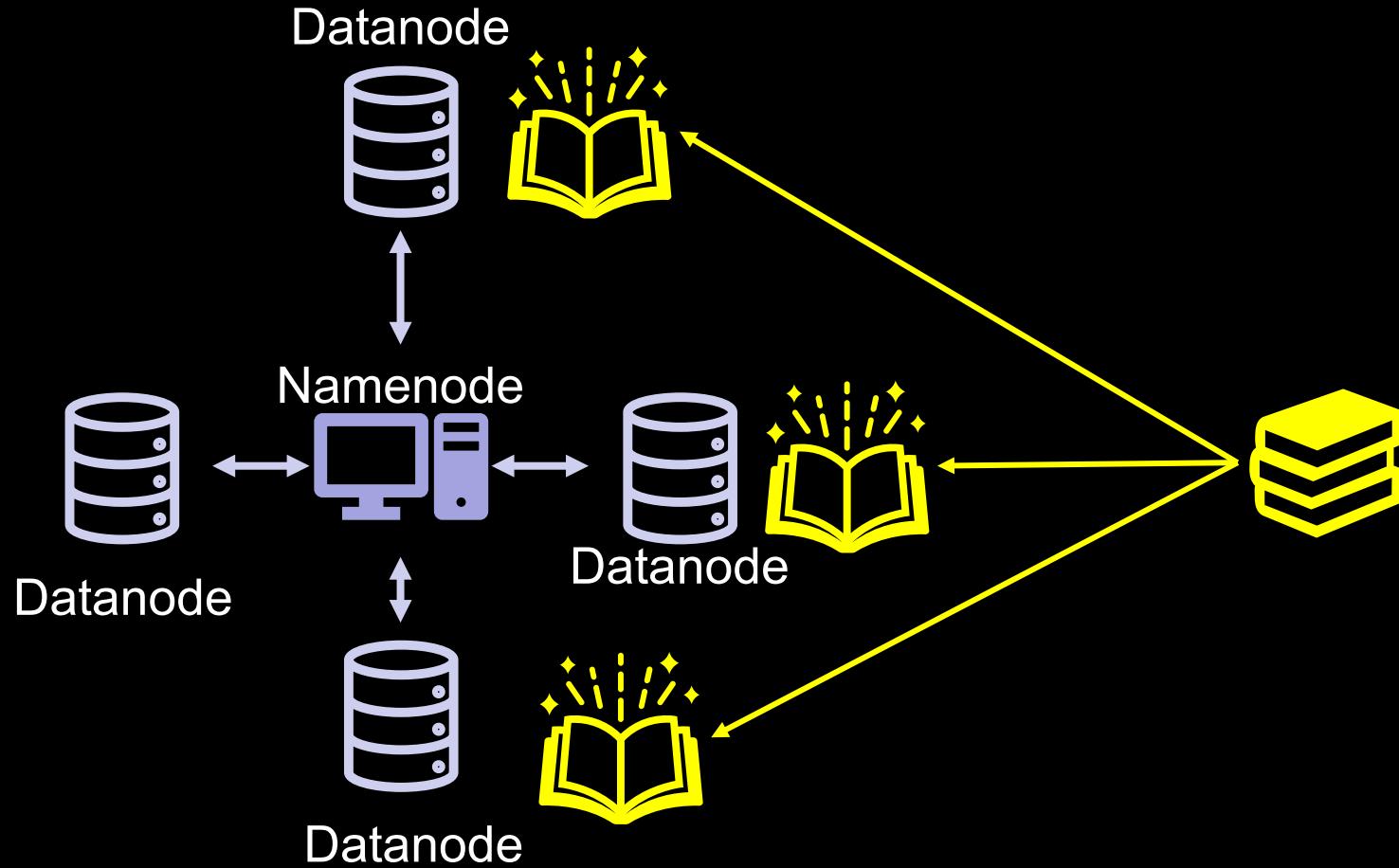
Parallelizzazione dei lavori in mapReduce



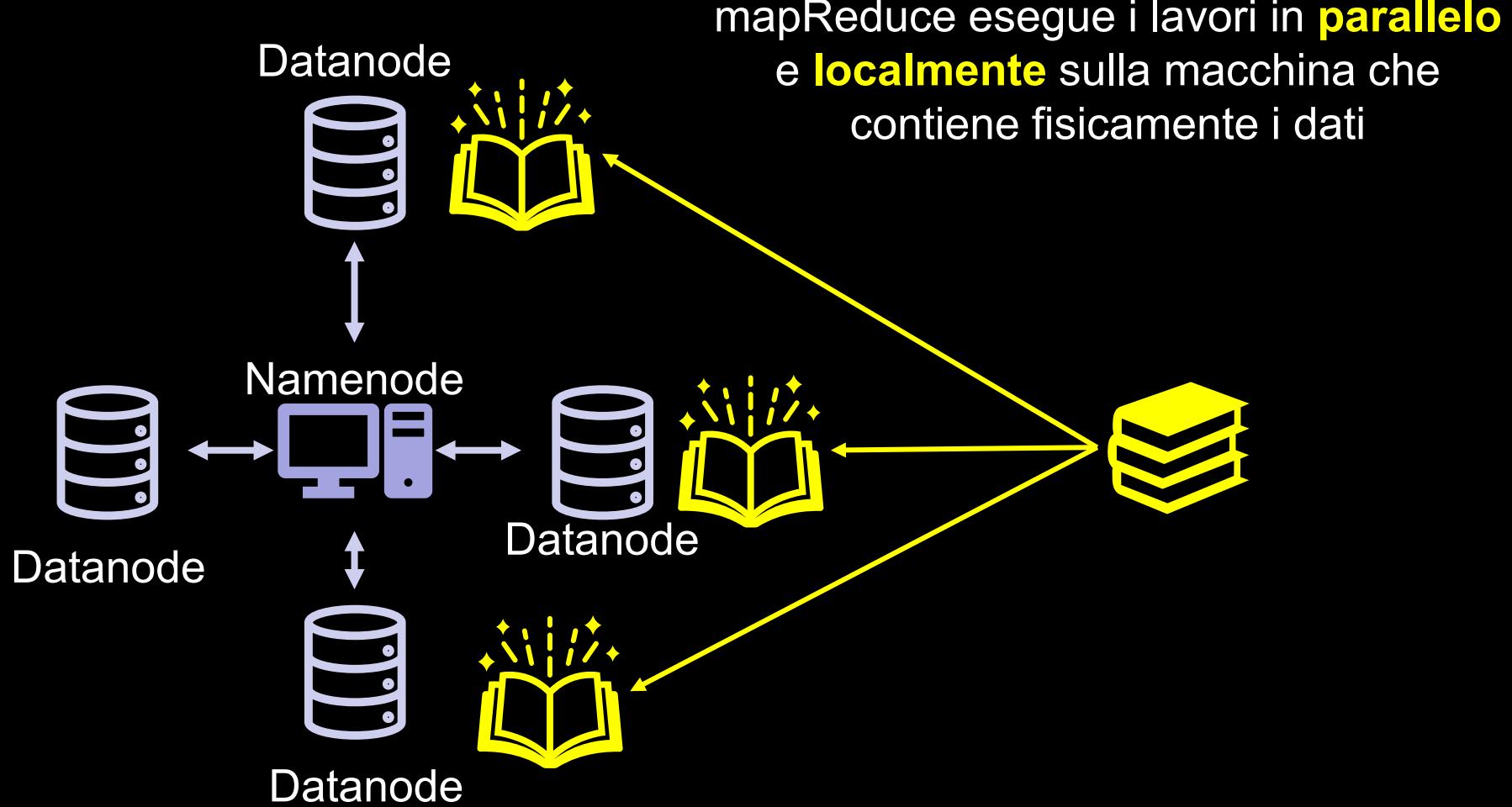
Parallelizzazione dei lavori in mapReduce



Parallelizzazione dei lavori in mapReduce

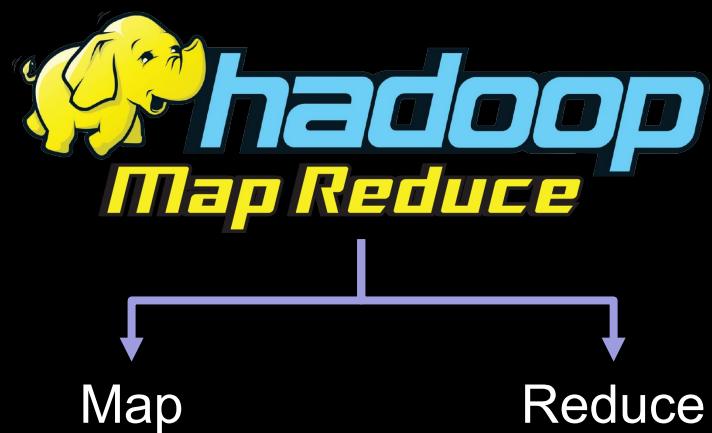


Parallelizzazione dei lavori in mapReduce



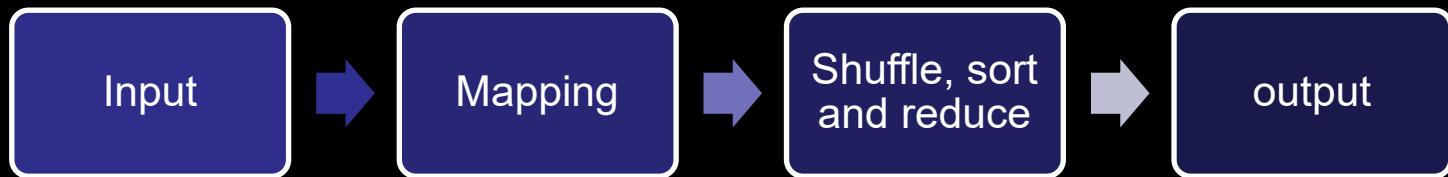
Componenti

MapReduce è un motore di processo che analizza una grande quantità di dati in forma distribuita e in parallelo attraverso i processi di mapping e riduzione

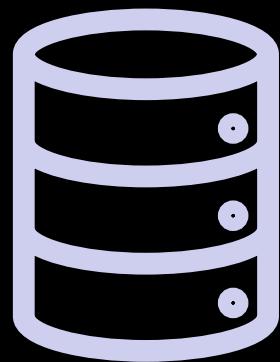


Processo

MapReduce è un motore di processo che analizza una grande quantità di dati in forma distribuita e in parallelo attraverso i processi di mapping e riduzione

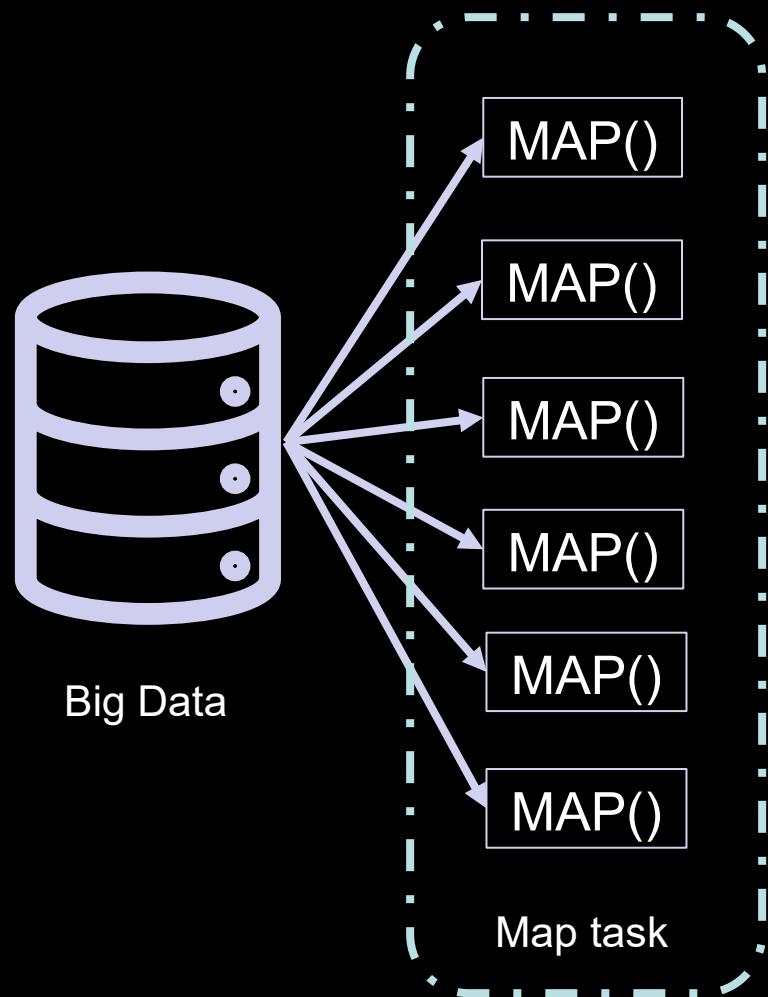


Processo

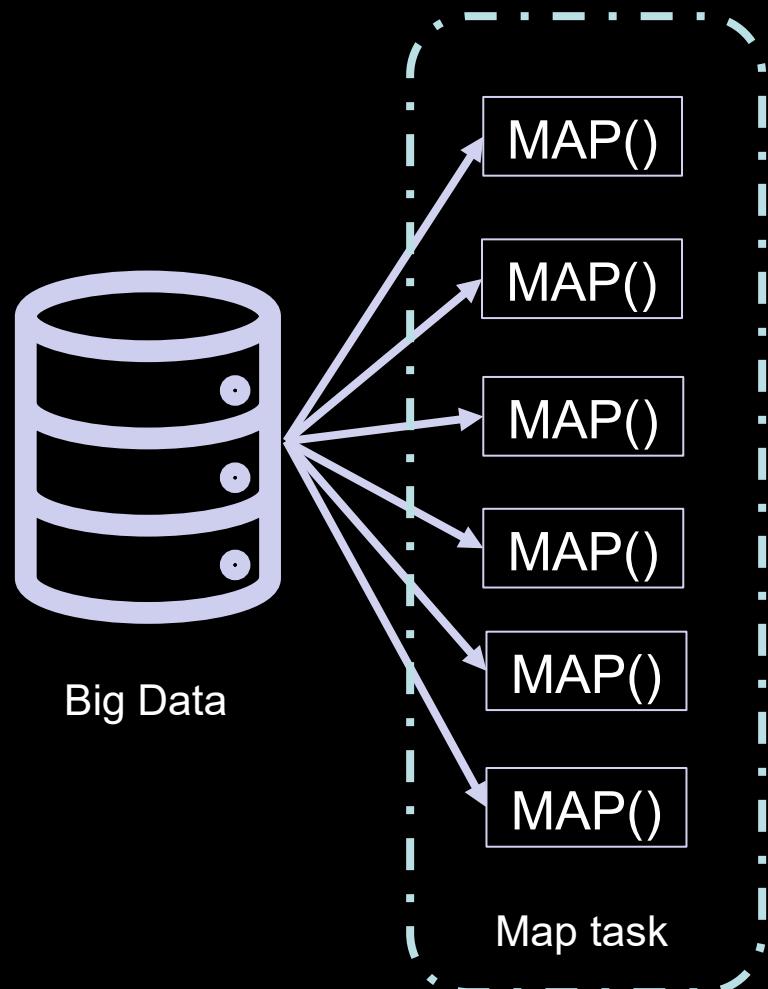


Big Data

Processo

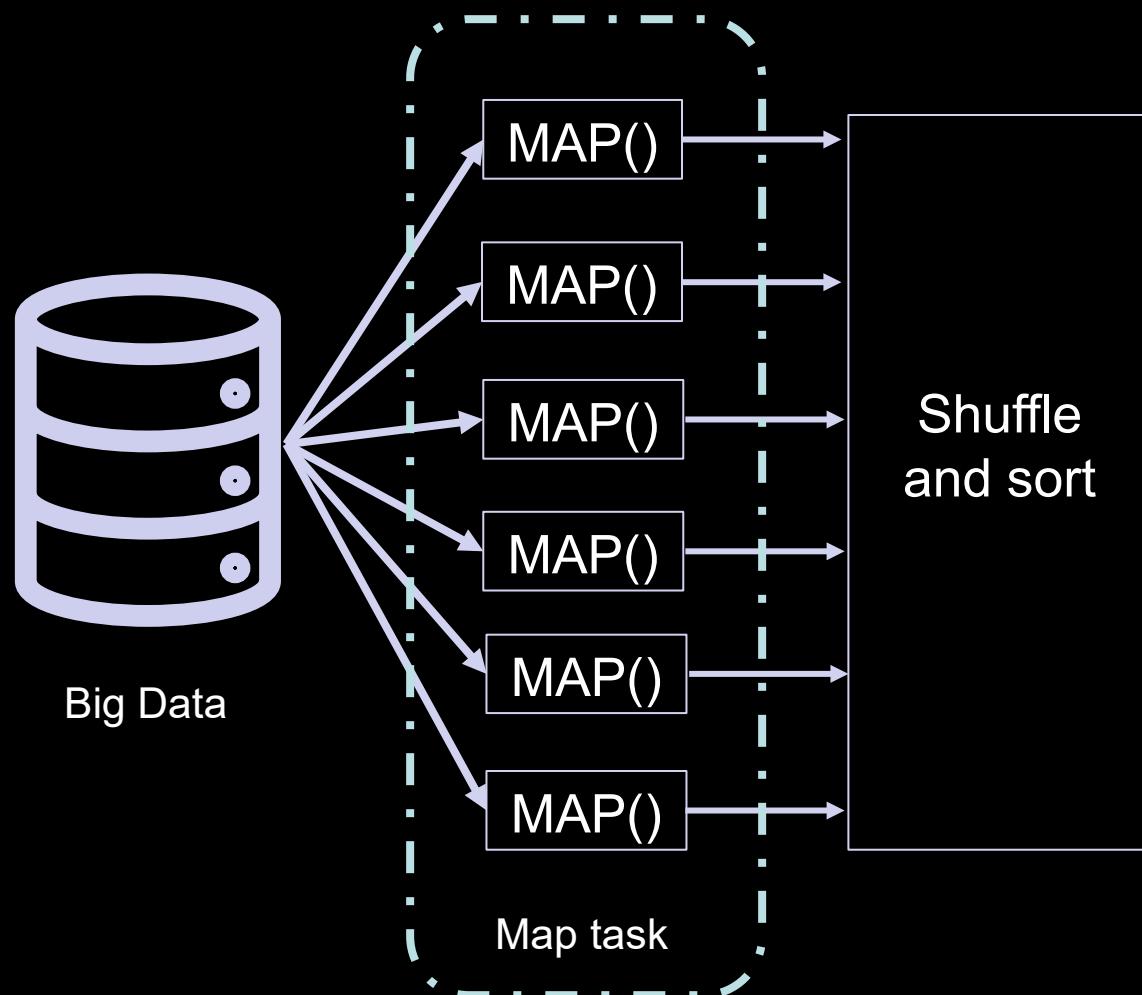


Processo

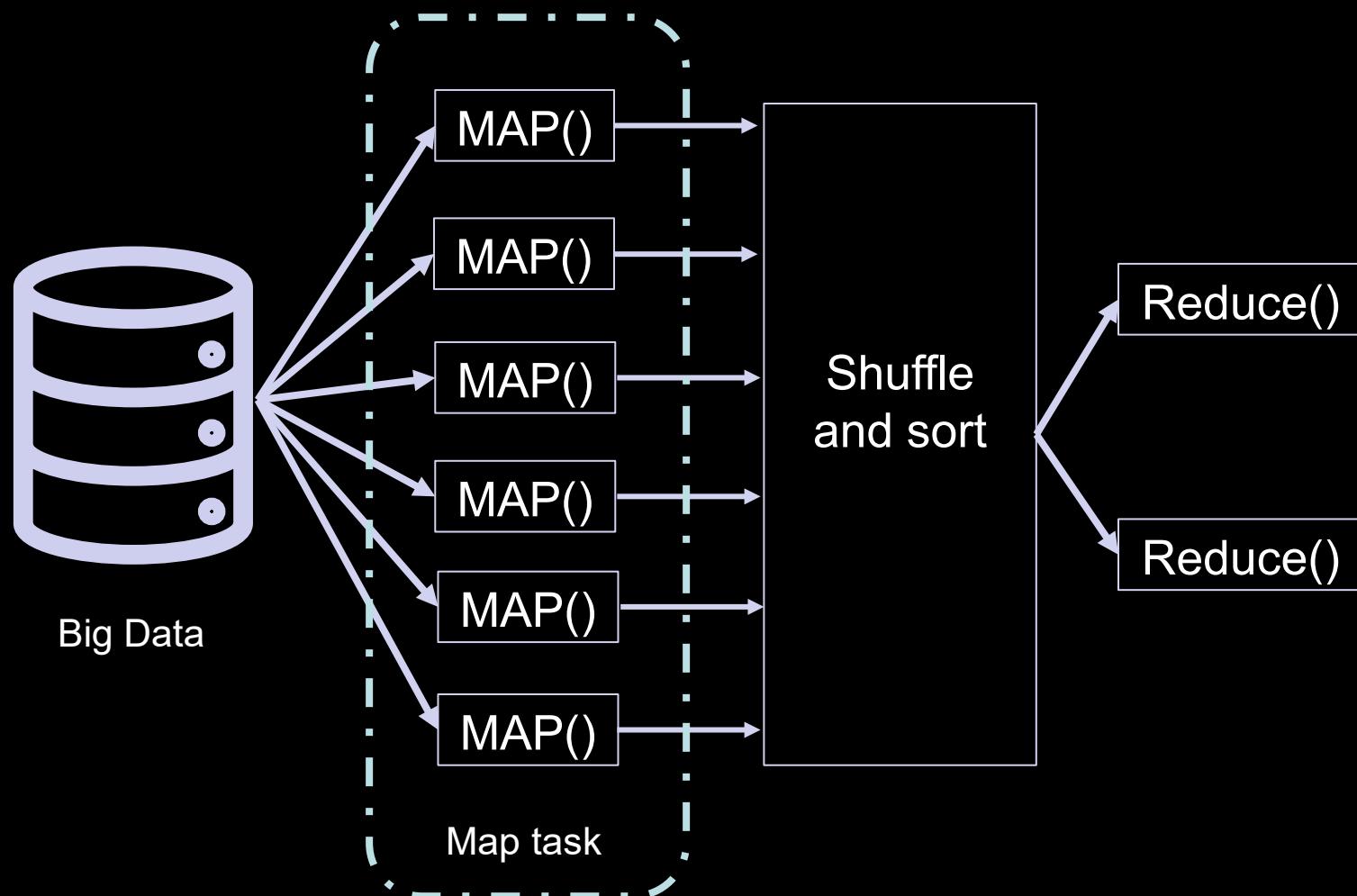


Dati distribuiti su datanodes allocati su rack diversi eseguono in parallelo diversi map task specificati dallo sviluppatore

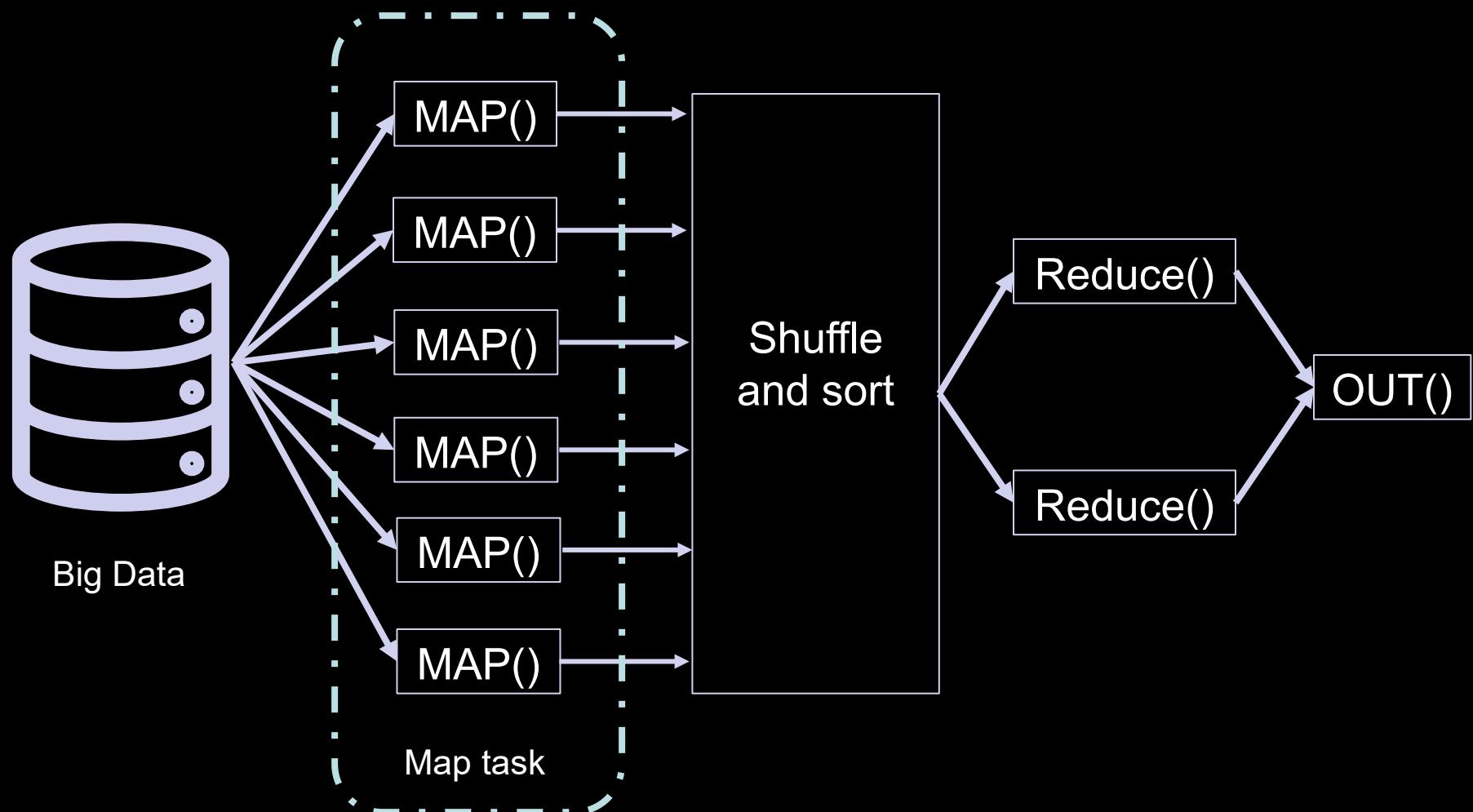
Data processing - MapReduce



Data processing - MapReduce



Data processing - MapReduce



Mapreduce job

MapReduce job è una unità di lavoro del client.

Consiste in:

- input data
- il codice map reduce
- informazioni di configurazione

Mapreduce job

MapReduce job è una unità di lavoro del client.

Consiste in:

- input data
- il codice map reduce
- informazioni di configurazione

Hadoop divide gli input di un task mapreduce in piccolo pezzi chiamati “**input splits**”.

Mapreduce job

MapReduce job è una unità di lavoro del client.

Consiste in:

- input data
- il codice map reduce
- informazioni di configurazione

Hadoop divide gli input di un task mapreduce in piccolo pezzi chiamati “**input splits**”.

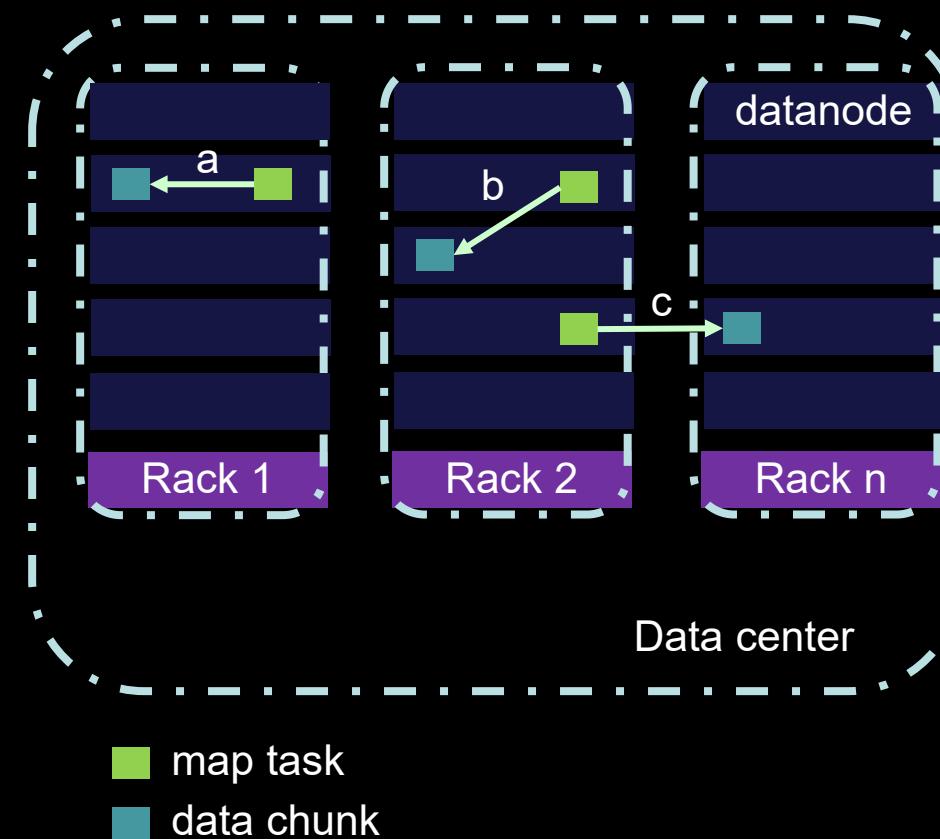
Piccoli split causano un overhead delle macchine

Grandi split tendono a caricare computazionalmente pochi nodi

Mapping jobs – ottimizzazione dati locale

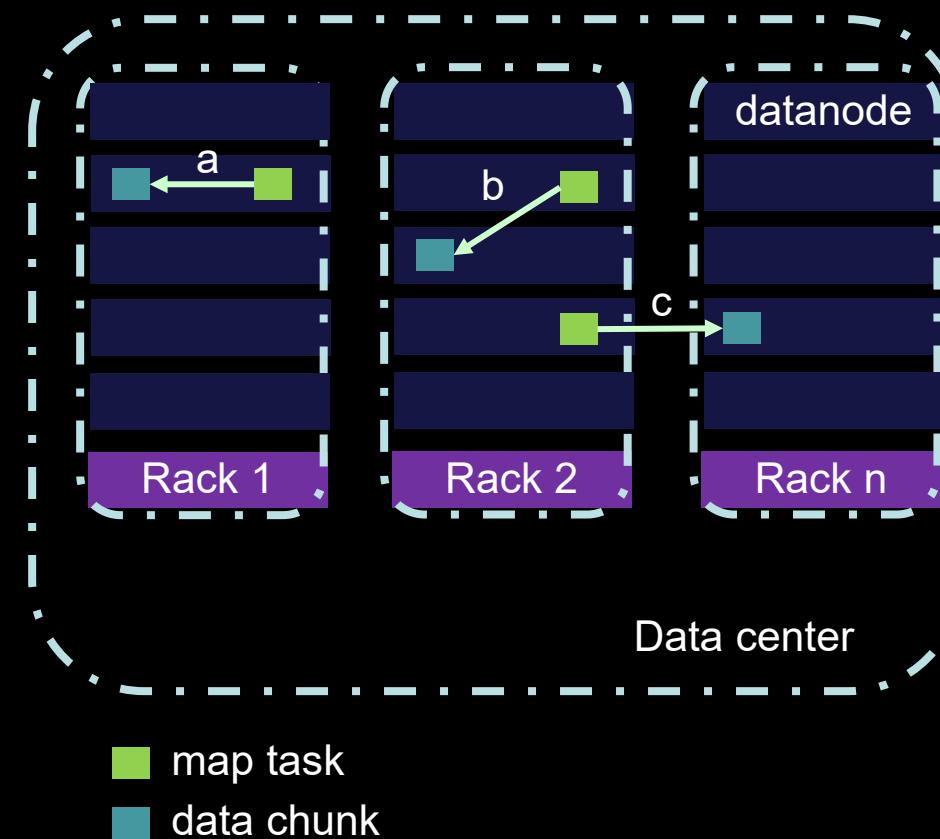
Un mapping job fa il suo meglio quando:

- La dimensione dei chunks è ben bilanciata (default 128MB)
- I jobs possono essere lanciati in locale (dove i dati sono allocati)

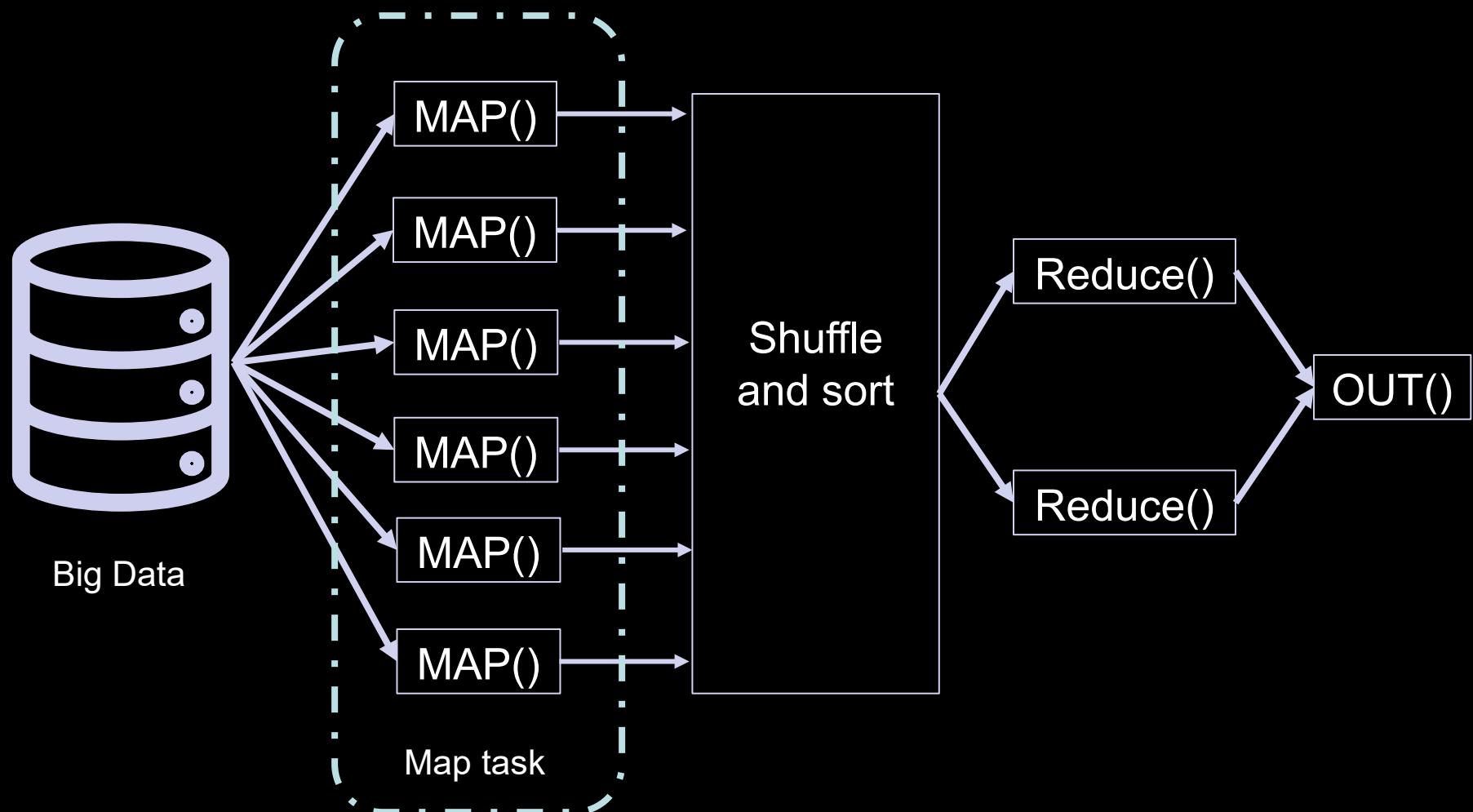


Mapping jobs – ottimizzazione dati locale

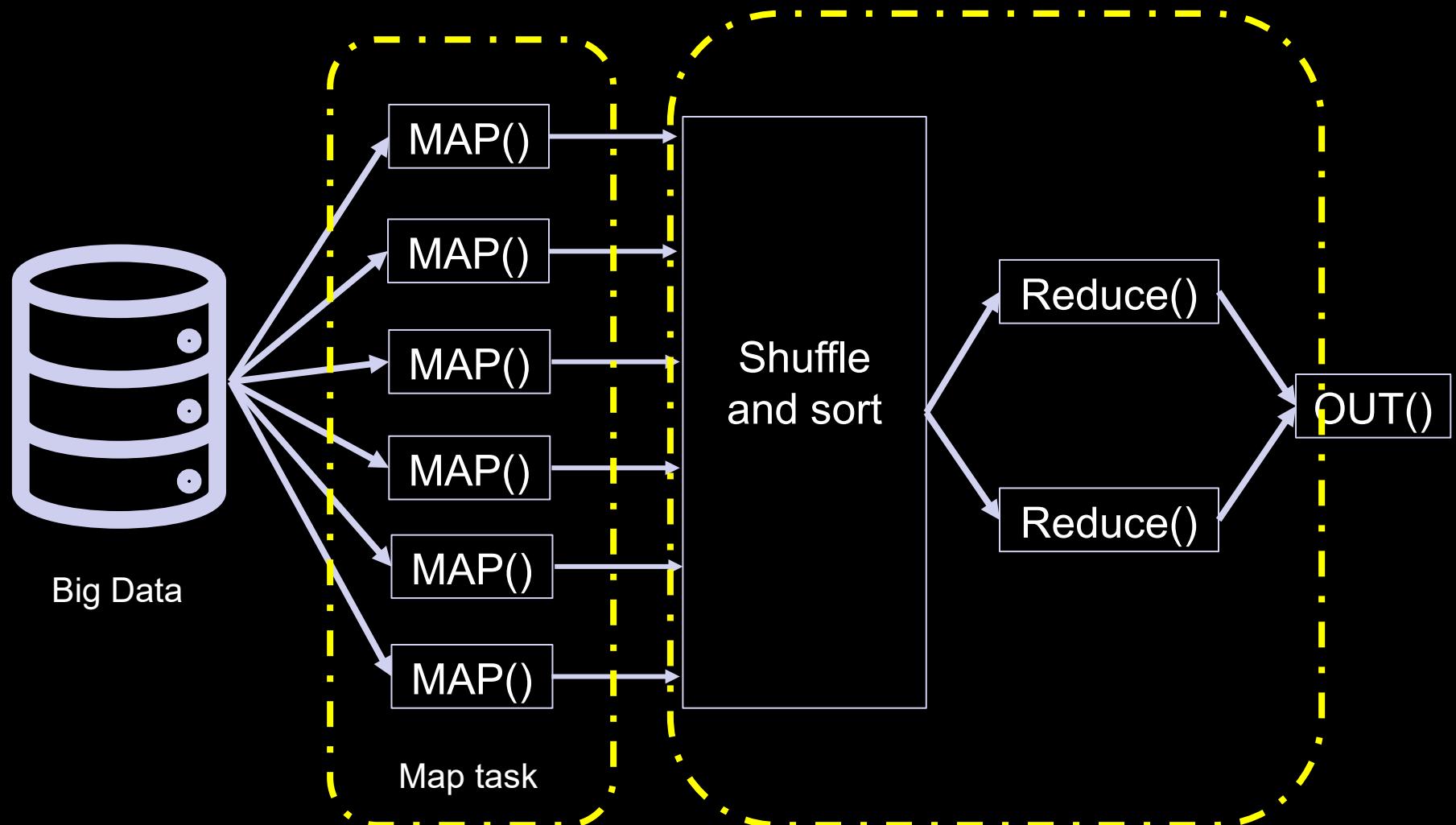
NOTA: Map task scrivono file temporanei in locale, non sul HDFS



Data processing - MapReduce



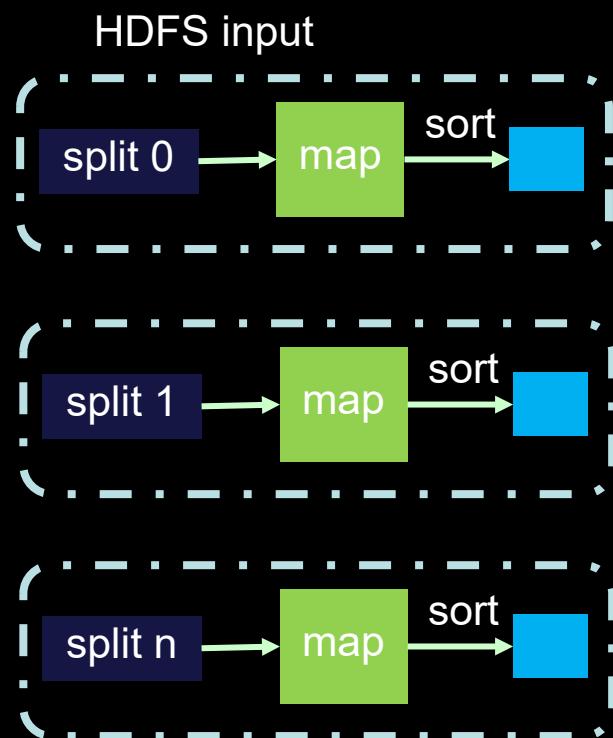
Data processing - MapReduce



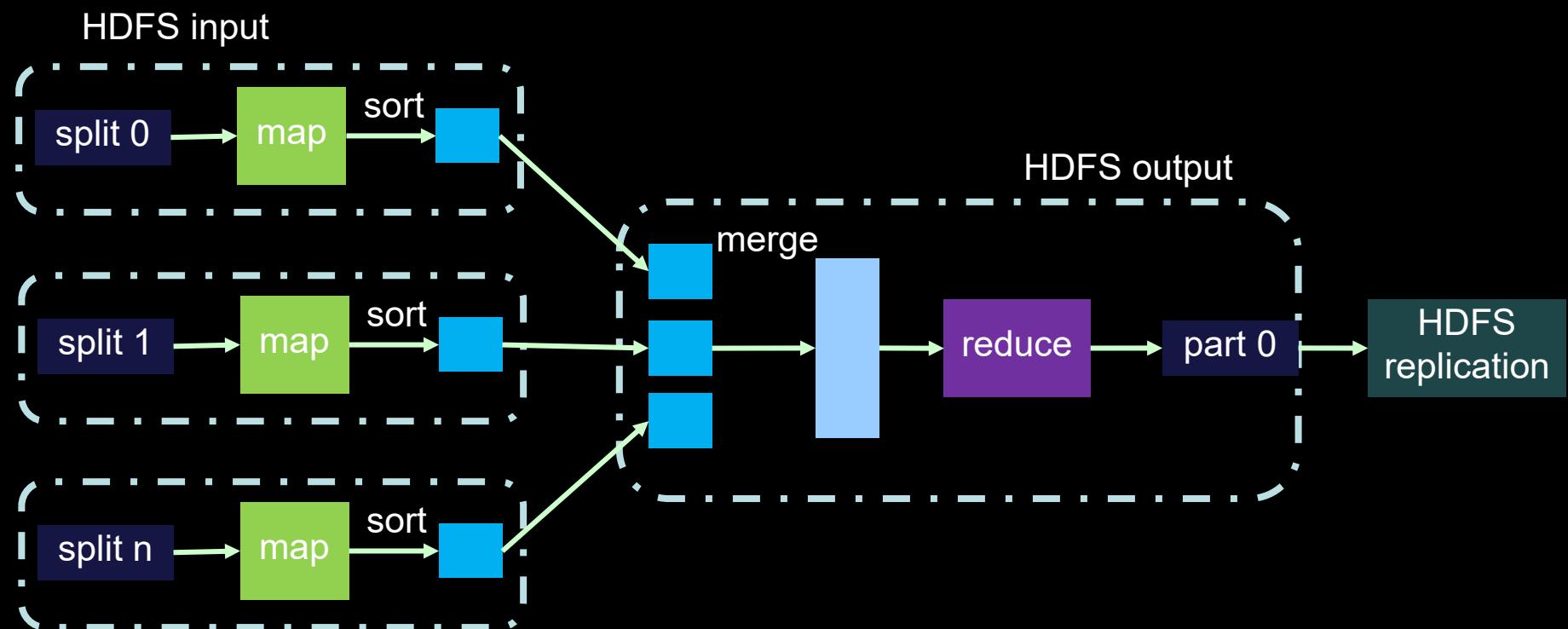
I map task **possono** sfruttare il principio di data locality

I reduce task **NON** possono sfruttare il principio di data locality

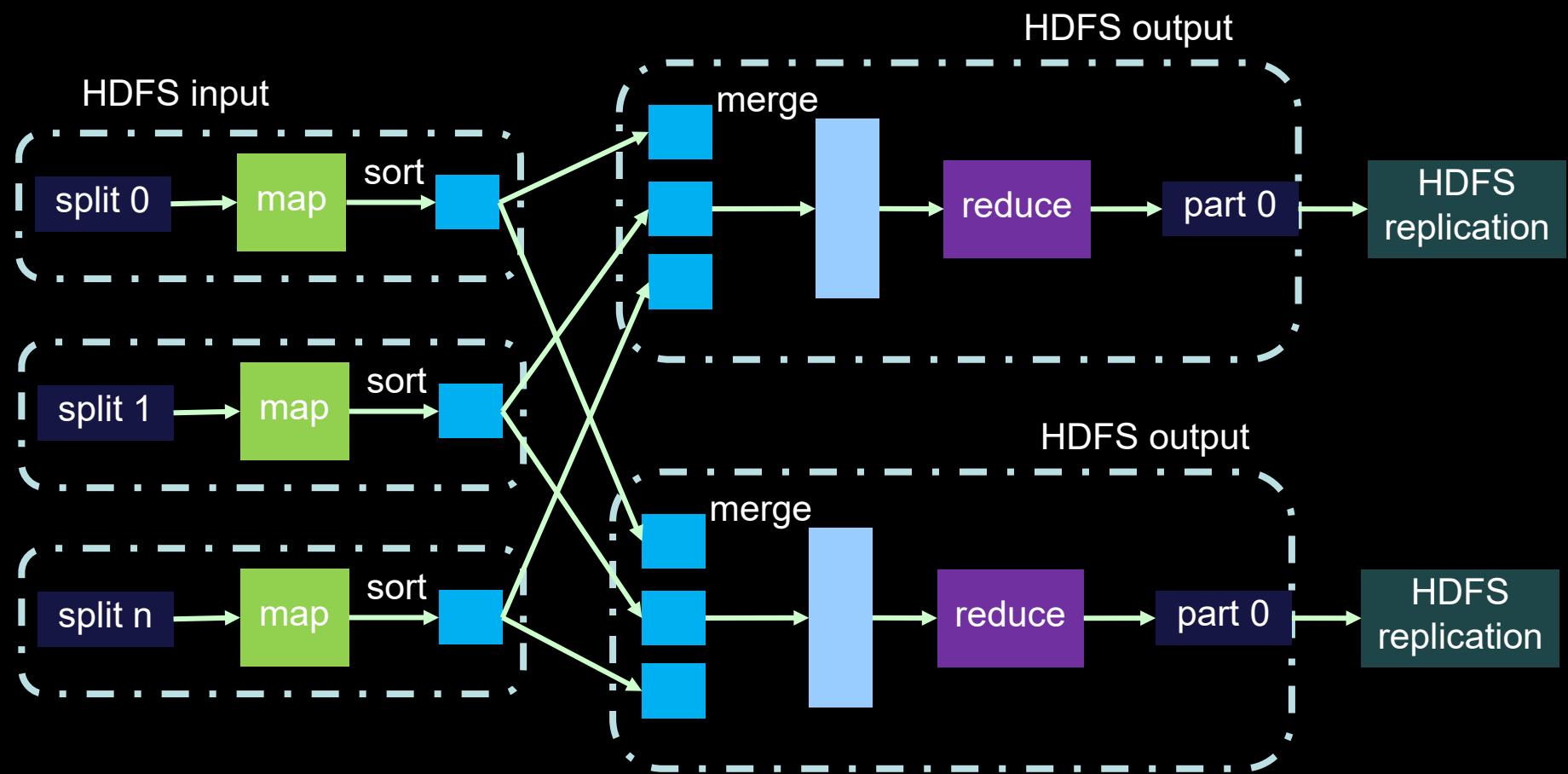
Mapreduce dataflow



Mapreduce dataflow



Mapreduce dataflow



Combinatori

I combinatori sono delle funzioni che combinano un output intermedio per ottimizzare la velocità dell'output finale

Combinatori

I combinatori sono delle funzioni che combinano un output intermedio per ottimizzare la velocità dell'output finale



Processa temperature nel 2001
da gennaio a giugno



Processa temperature nel 2001
da luglio a dicembre

Combinatori

I combinatori sono delle funzioni che combinano un output intermedio per ottimizzare la velocità dell'output finale



Processa temperature nel 2001
da gennaio a giugno

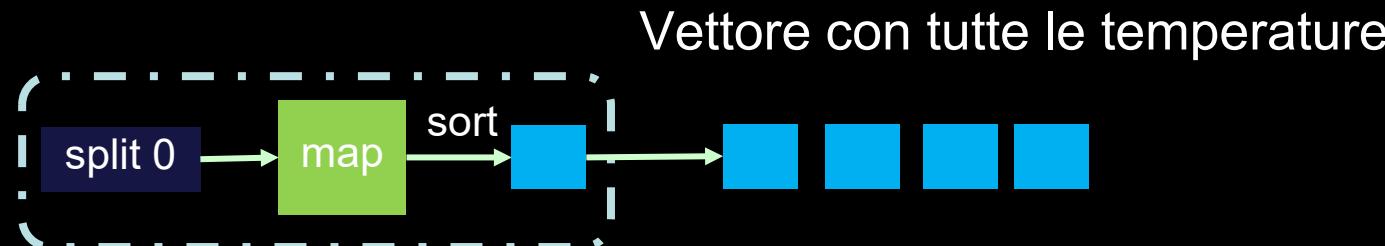
Supponiamo che la query sia
 $\max(2001)$



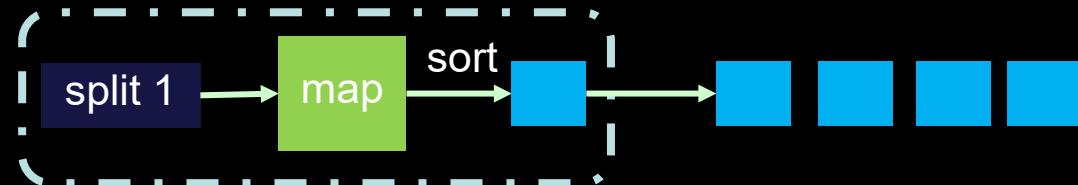
Processa temperature nel 2001
da luglio a dicembre

Combinatori

Query: Massima temperatura anno 2001



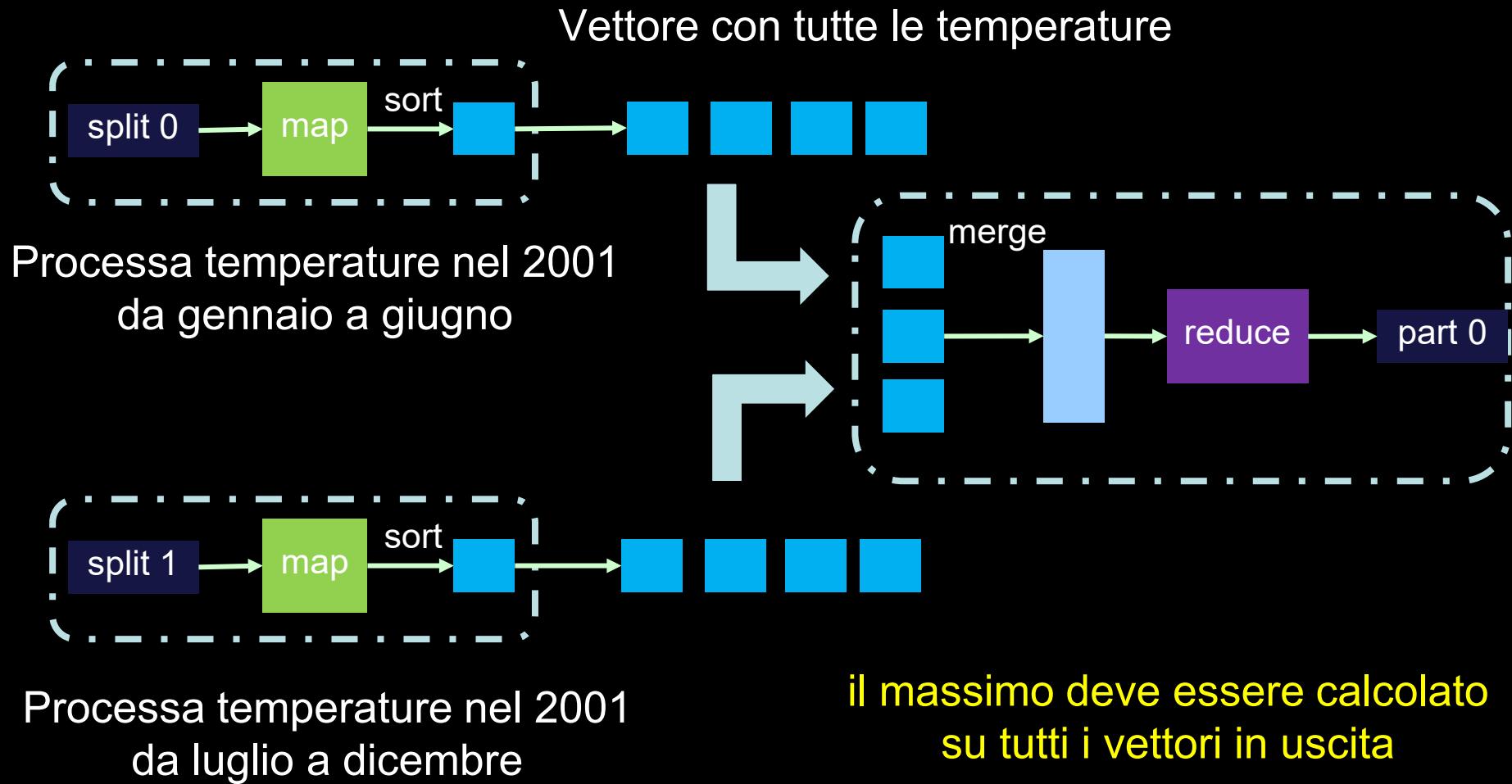
Processa temperature nel 2001
da gennaio a giugno



Processa temperature nel 2001
da luglio a dicembre

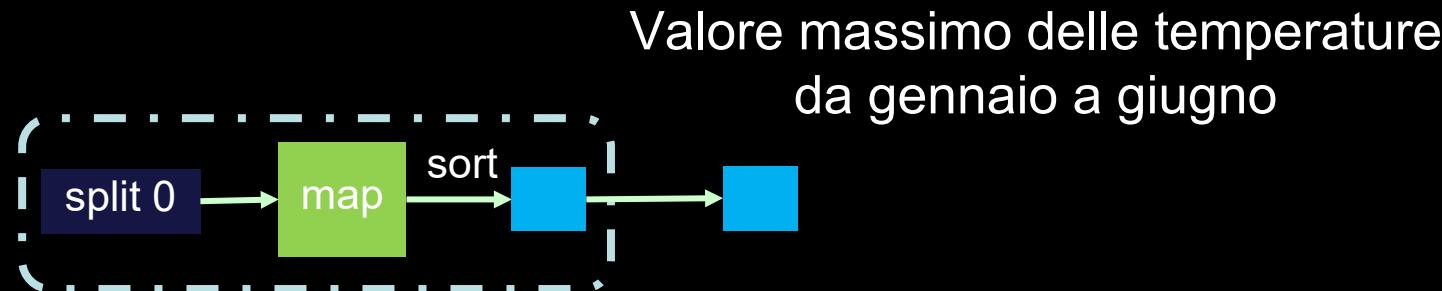
Combinatori

Query: Massima temperatura anno 2001

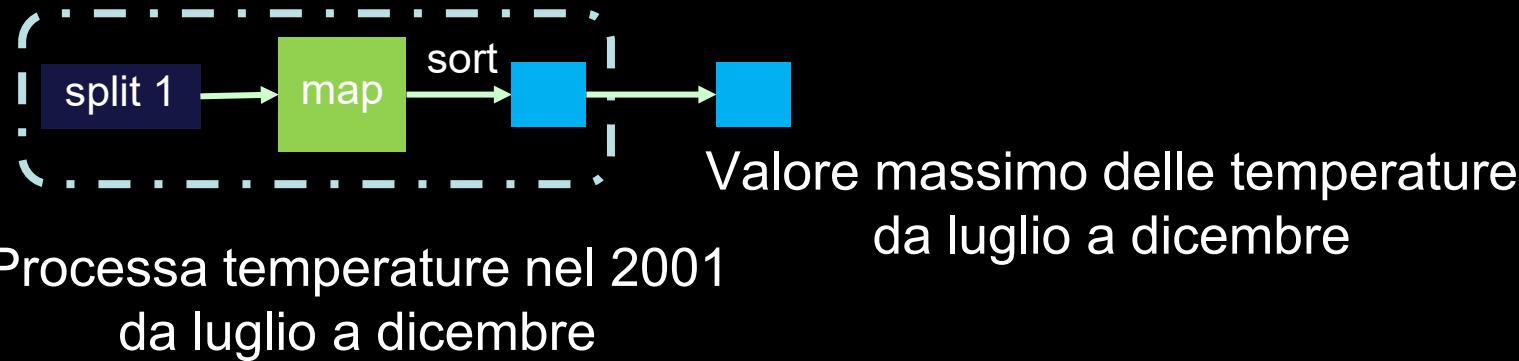


Combinatori

Query: Massima temperatura anno 2001



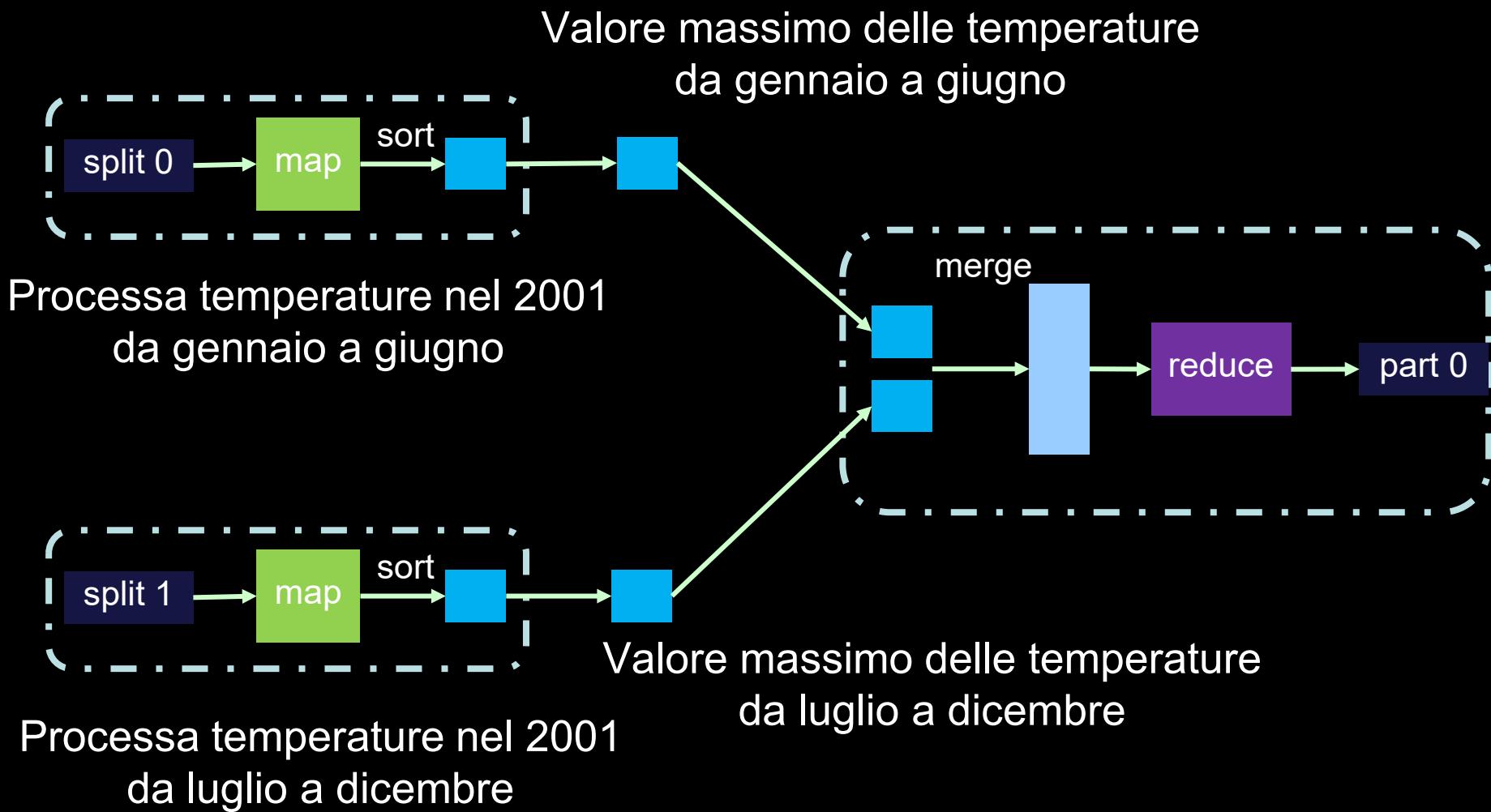
Processa temperature nel 2001
da gennaio a giugno



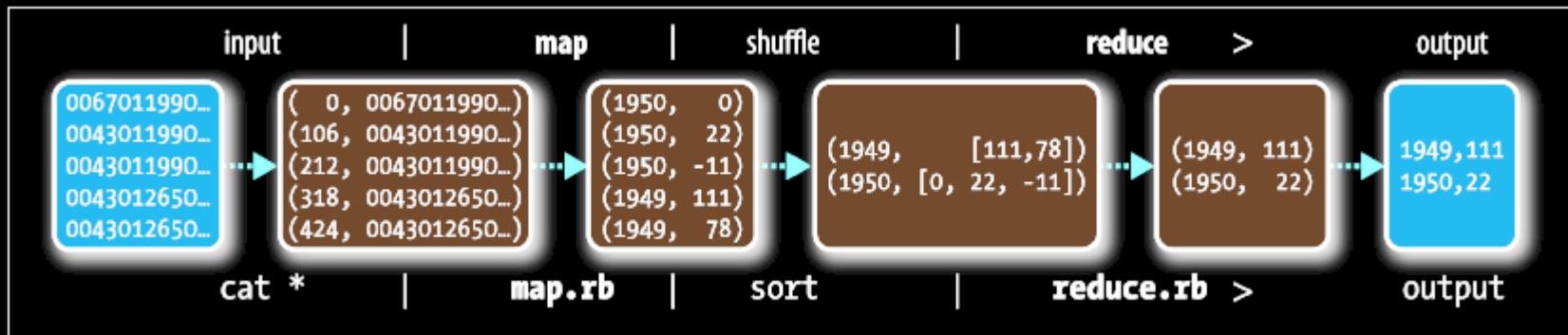
Processa temperature nel 2001
da luglio a dicembre

Combinatori

Query: Massima temperatura anno 2001



Esempio – Dal libro “Hadoop”



Esempio – Dal libro “Hadoop”

input

map

shuffle

reduce

>

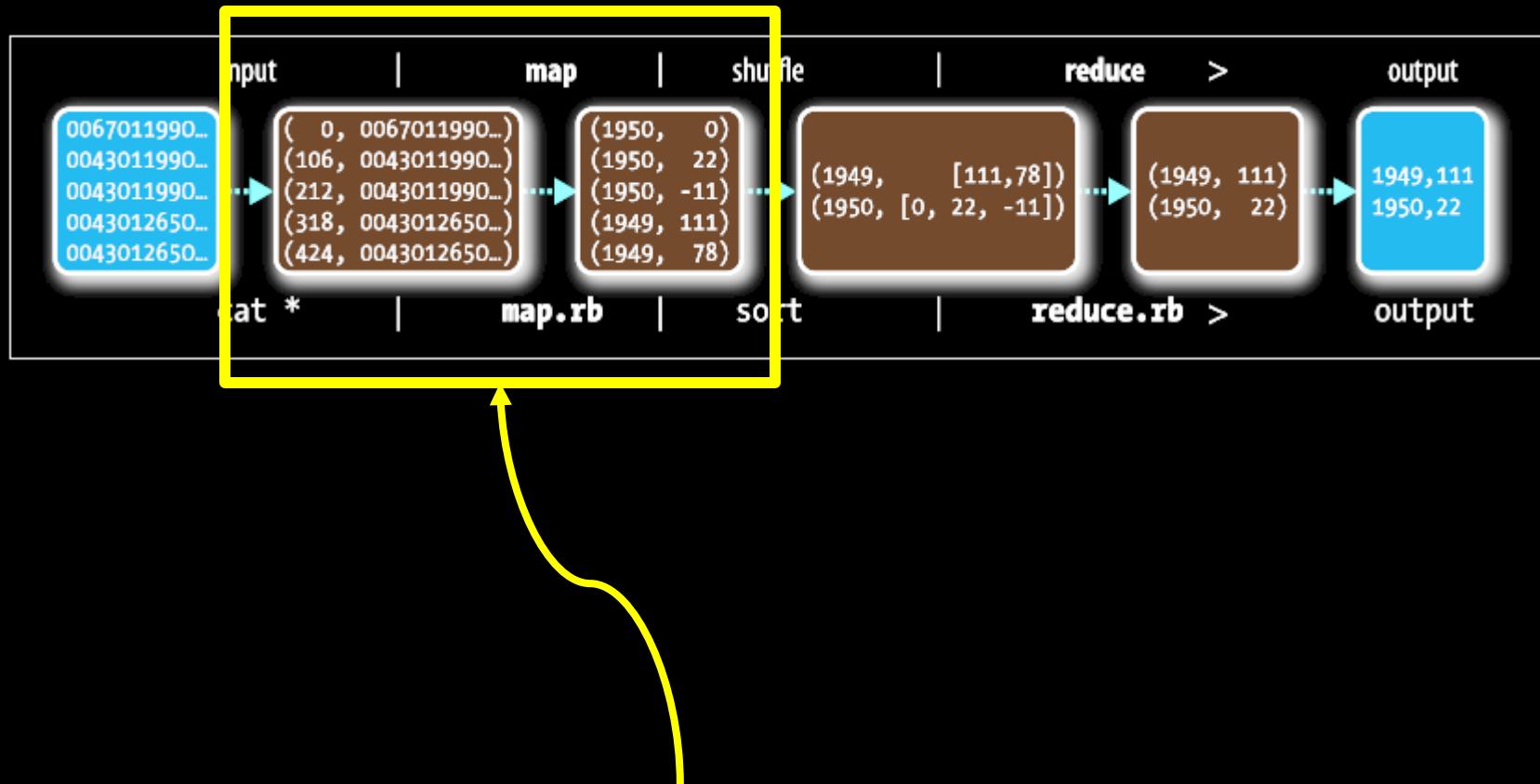
output

Example 2-1. Format of a National Climatic Data Center record

```
006701199  
004301199 0057  
004301199 332130 # USAF weather station identifier  
004301265 99999 # WBAN weather station identifier  
004301265 19500101 # observation date  
0300 # observation time  
4  
+51317 # latitude (degrees x 1000)  
+028783 # longitude (degrees x 1000)  
FM-12  
+0171 # elevation (meters)  
99999  
V020  
320 # wind direction (degrees)  
1 # quality code  
N  
0072  
1  
00450 # sky ceiling height (meters)  
1 # quality code  
C  
N  
010000 # visibility distance (meters)  
1 # quality code  
N  
9  
-0128 # air temperature (degrees Celsius x 10)  
1 # quality code  
-0139 # dew point temperature (degrees Celsius x 10)  
1 # quality code  
10268 # atmospheric pressure (hectopascals x 10)  
1 # quality code
```

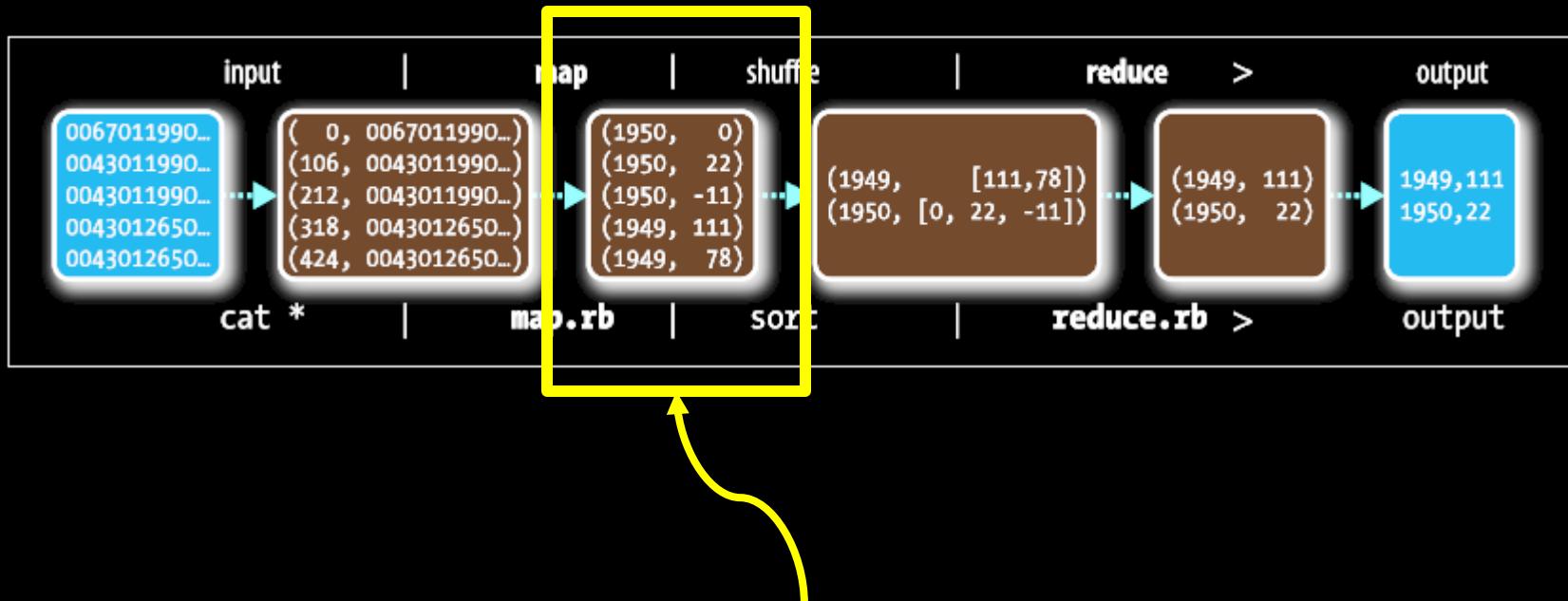


Esempio – Dal libro “Hadoop”



Nella prima fase di mapping potenzialmente ogni file può essere processato da un datanode diverso

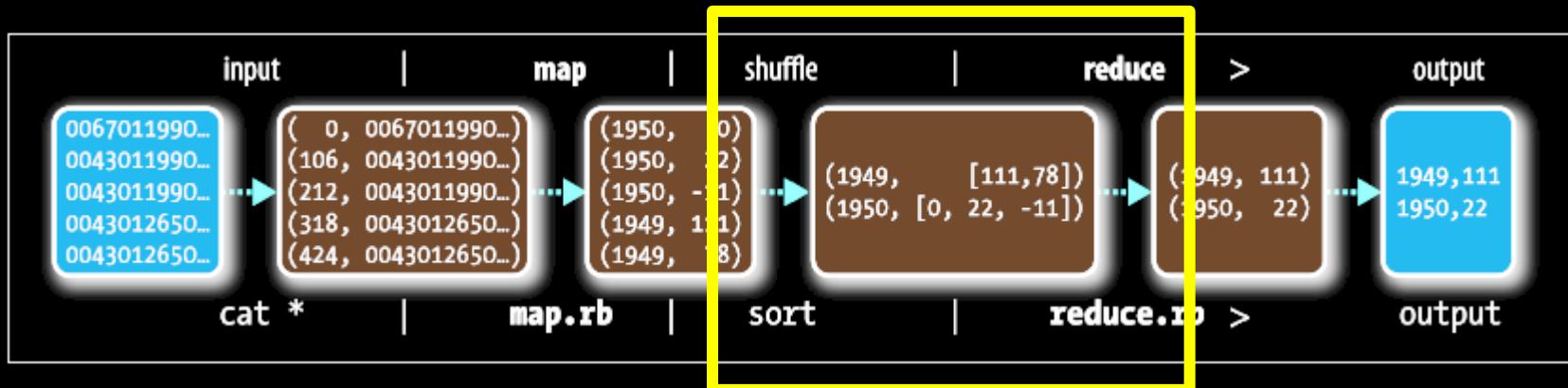
Esempio – Dal libro “Hadoop”



To visualize the way the map works, consider the following sample lines of input data (some unused columns have been dropped to fit the page, indicated by ellipses):

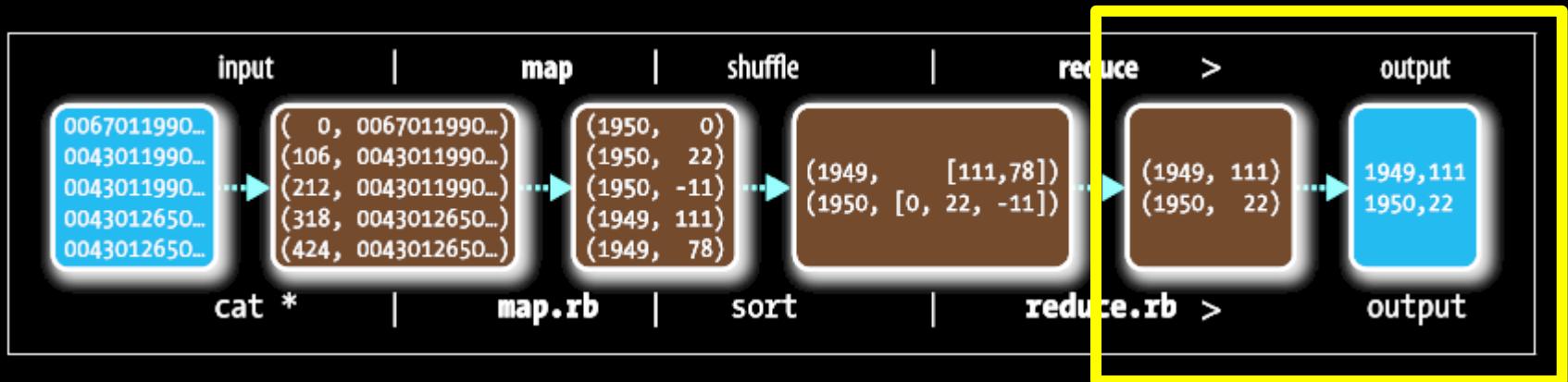
```
0067011990999991950051507004...9999999N9+00001+99999999999...  
0043011990999991950051512004...9999999N9+00221+99999999999...  
0043011990999991950051518004...9999999N9-00111+99999999999...  
0043012650999991949032412004...0500001N9+01111+99999999999...  
0043012650999991949032418004...0500001N9+00781+99999999999...
```

Esempio – Dal libro “Hadoop”



L'output per la funzione di mappatura è processato dal sistema di mapReduce prima di essere inviato alla funzione di riduzione.
Questo processo ordina e raggruppa per voce o chiave.

Esempio – Dal libro “Hadoop”



Da notare che mentre il primo processo può essere fatto in parallelo, questo processo deve essere fatto da una singola macchina

Tuttavia, tutto ciò che la funzione di riduzione deve fare è iterare sulle singole voci per trovare le informazioni richieste (il massimo)



Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

BIG DATA e Analisi dei Dati

Lezione 2.2 – YARN

Mauro Bellone,
Robotics and AI researcher

bellonemauro@gmail.com
www.maurobellone.com

Obiettivo

- ✓ Comprensione delle basi dell'architettura di YARN

YARN



Yet Another Resource Negotiator (YARN)

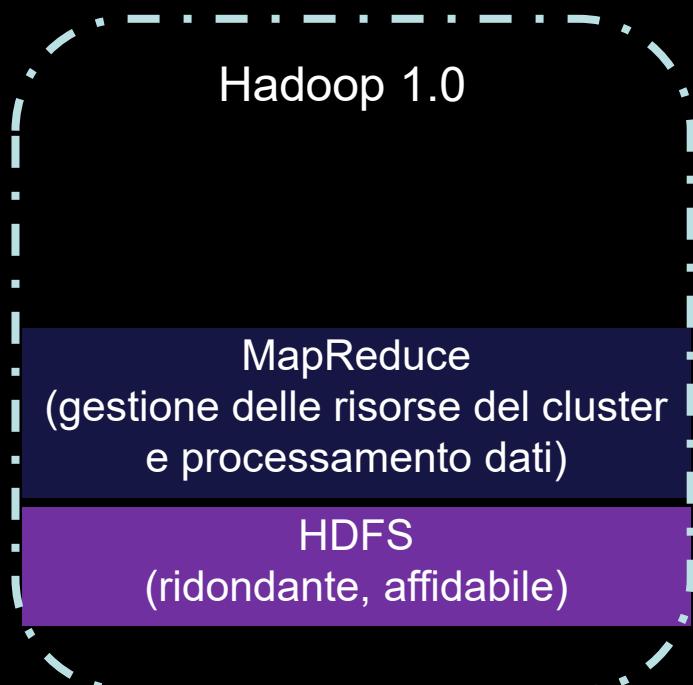
- YARN è un resource manager
- è stato creato separando il motore di processo e di gestione di mapReduce
- Monitora e gestisce i carico computazionale, mantiene un ambiente multiutente, gestisce la disponibilità delle risorse di Hadoop e implementa i controlli di sicurezza

YARN – cenno storico



< 2012

Gli utenti potevano scrivere
mapReduce programs using
scripting languages (java,
python)



YARN – cenno storico

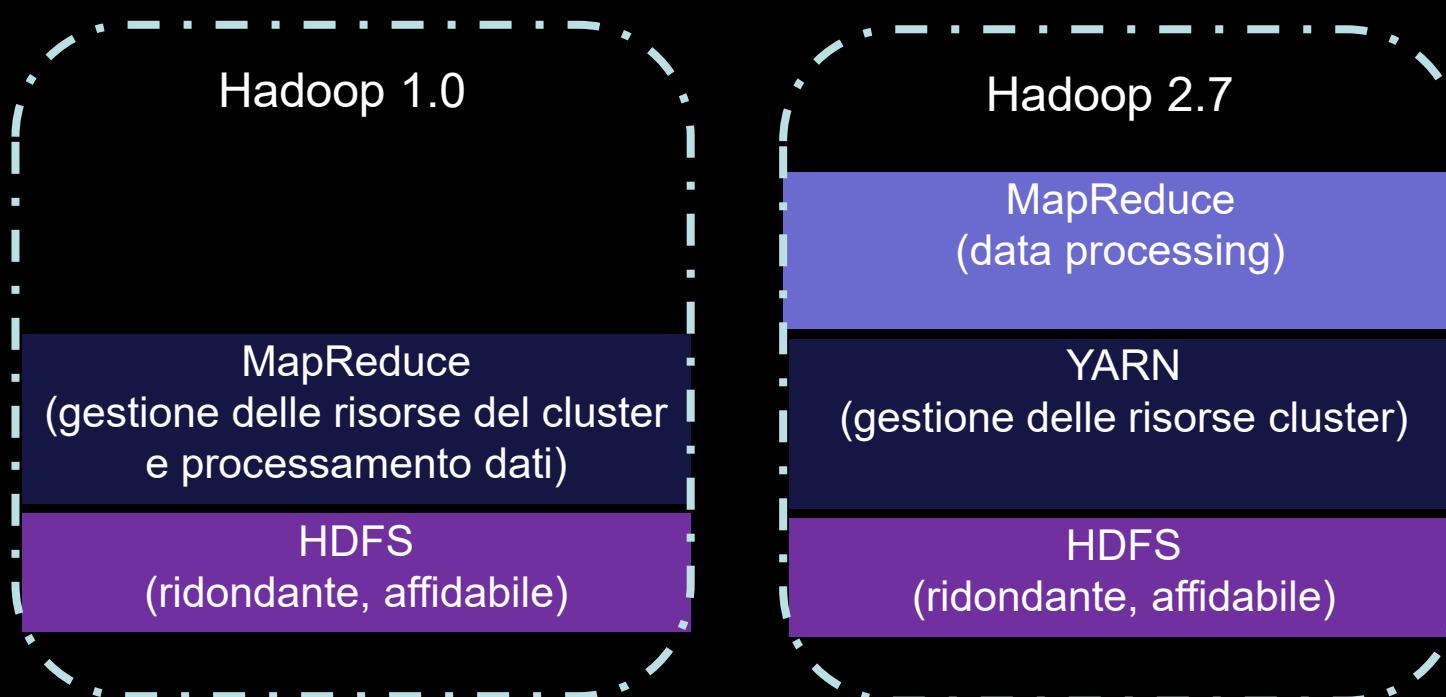


< 2012

Gli utenti potevano scrivere mapReduce programs using scripting languages (java, python)

> 2012

Gli utenti possono lavorare su modelli di processo multipli in aggiunta a mapReduce



Yahoo – Use case

- Yahoo è stata la prima azienda ad adottare Hadoop in produzione e nel 2012 ha avuto delle difficoltà a mantenere il flusso di processo a causa delle limitazioni intriseche di mapreduce.

Yahoo – Use case

- Yahoo è stata la prima azienda ad adottare Hadoop in produzione e nel 2012 ha avuto delle difficoltà a mantenere il flusso di processo a causa delle limitazioni intriseche di mapreduce.
- Dopo l'implementazione di YARN nel 2013 ha installato 30000 nodi di produzione su
 - Spark per i processi iterative
 - Storm per i processi di streaming
 - Hadoop per il processamento di batch

Yahoo – Use case

- Yahoo è stata la prima azienda ad adottare Hadoop in produzione e nel 2012 ha avuto delle difficoltà a mantenere il flusso di processo a causa delle limitazioni intriseche di mapreduce.
- Dopo l'implementazione di YARN nel 2013 ha installato 30000 nodi di produzione su
 - Spark per i processi iterative
 - Storm per i processi di streaming
 - Hadoop per il processamento di batch
- Questa soluzione è stata possibile solo grazie a YARN e l'implementazione di frameworks di processo multipli

Scheduling policy

Hardware:

- MAX throughput
- MAX CPU/GPU usage etc.
- MIN memory access

Software:

- FCFS First come first served o anche First-in first-out (FIFO)
- Fixed priority
- Round robin
- Time table-driven
- Dynamic priority scheduling
 - Earliest deadline first (EDF)
 - Least laxity first

Scheduling policy

Hardware:

- MAX throughput
- MAX CPU/GPU usage etc.
- MIN memory access

Software:

- FCFS First come first served o anche First-in first-out (FIFO)
- Fixed priority
- Round robin
- Time table-driven
- Dynamic priority scheduling
 - Earliest deadline first (EDF)
 - Least laxity first

Scheduling policy

Hardware:

- MAX throughput
- MAX CPU/GPU usage etc.
- MIN memory access

Software:

- FCFS First come first served o anche First-in first-out (FIFO)
- Fixed priority
- Round robin
- Time table-driven
- Dynamic priority scheduling
 - Earliest deadline first (EDF)
 - Least laxity first

YARN

YARN è un resource manager che lavora su un singolo cluster offrendo una serie di vantaggi dalla sua prospettiva di alto livello quali:

YARN

YARN è un resource manager che lavora su un singolo cluster offrendo una serie di vantaggi dalla sua prospettiva di alto livello quali:

- ✓ Massimizzazione dell'uso del cluster

Le risorse che non sono usate da un processo
possono essere usate da un altro

YARN

YARN è un resource manager che lavora su un singolo cluster offrendo una serie di vantaggi dalla sua prospettiva di alto livello quali:

- ✓ Massimizzazione dell'uso del cluster
- ✓ Bassi costi operazionali

Bisogna gestire solo un cluster

YARN

YARN è un resource manager che lavora su un singolo cluster offrendo una serie di vantaggi dalla sua prospettiva di alto livello quali:

- ✓ Massimizzazione dell'uso del cluster
- ✓ Bassi costi operazionali
- ✓ Reduced data motion

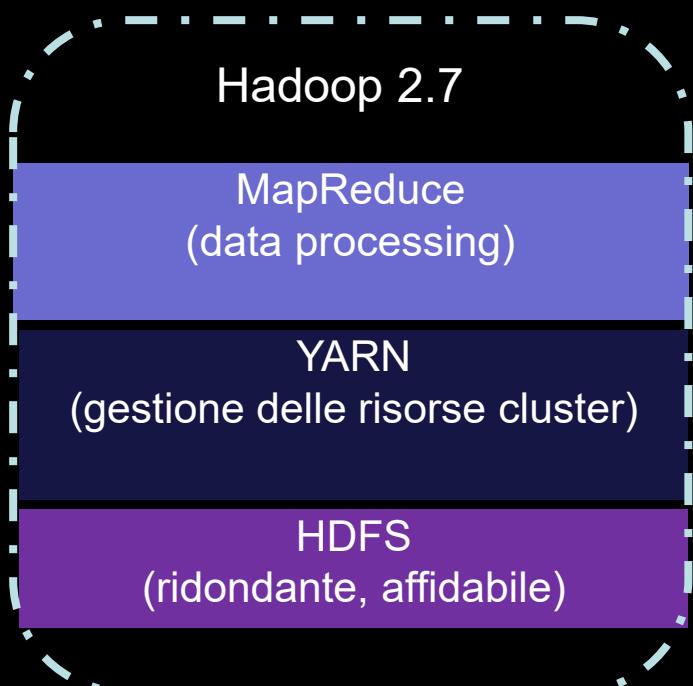
non è necessario spostare dati tra Hadoop YARN e gli altri sistemi che sono eseguiti su altri clusters

Infrastruttura di YARN

L'infrastruttura di YARN è responsabile di fornire le risorse computazionali per le applicazioni in esecuzione

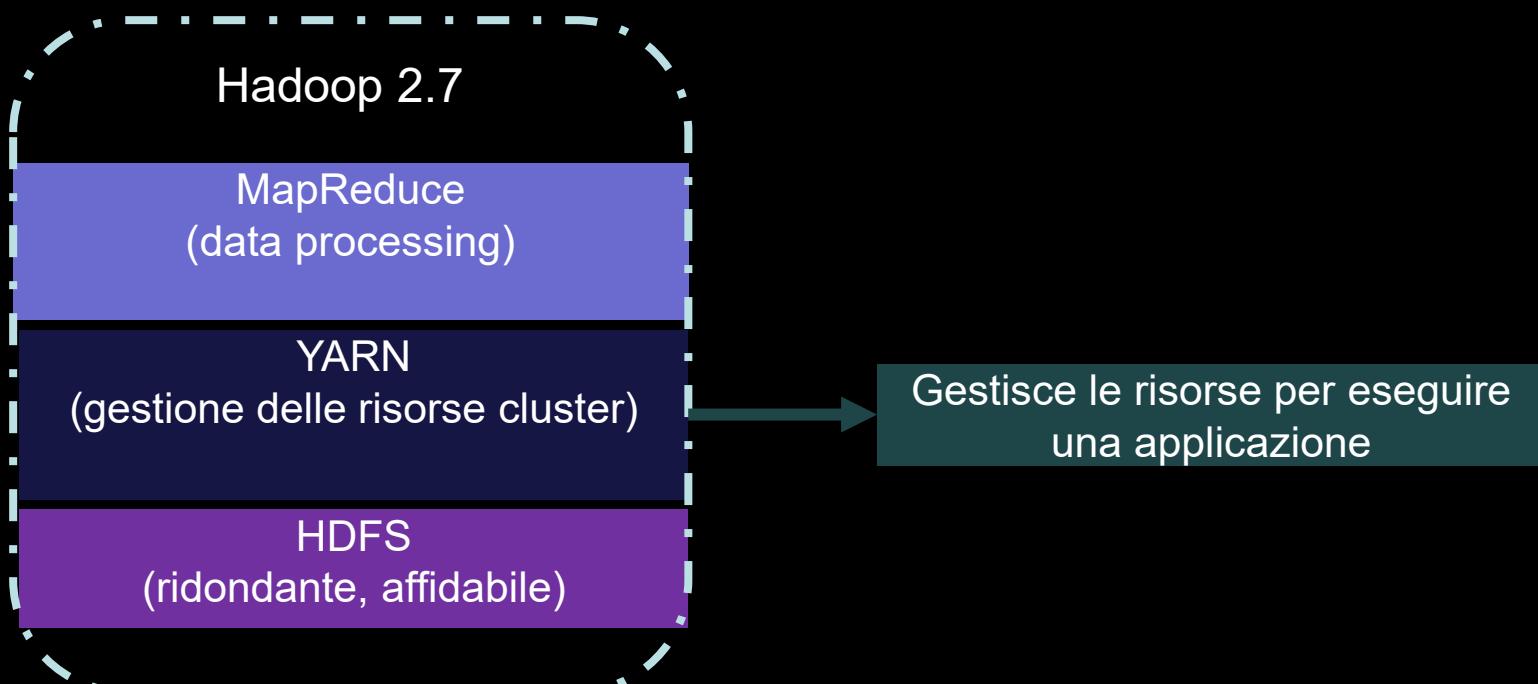
Infrastruttura di YARN

L'infrastruttura di YARN è responsabile di fornire le risorse computazionali per le applicazioni in esecuzione



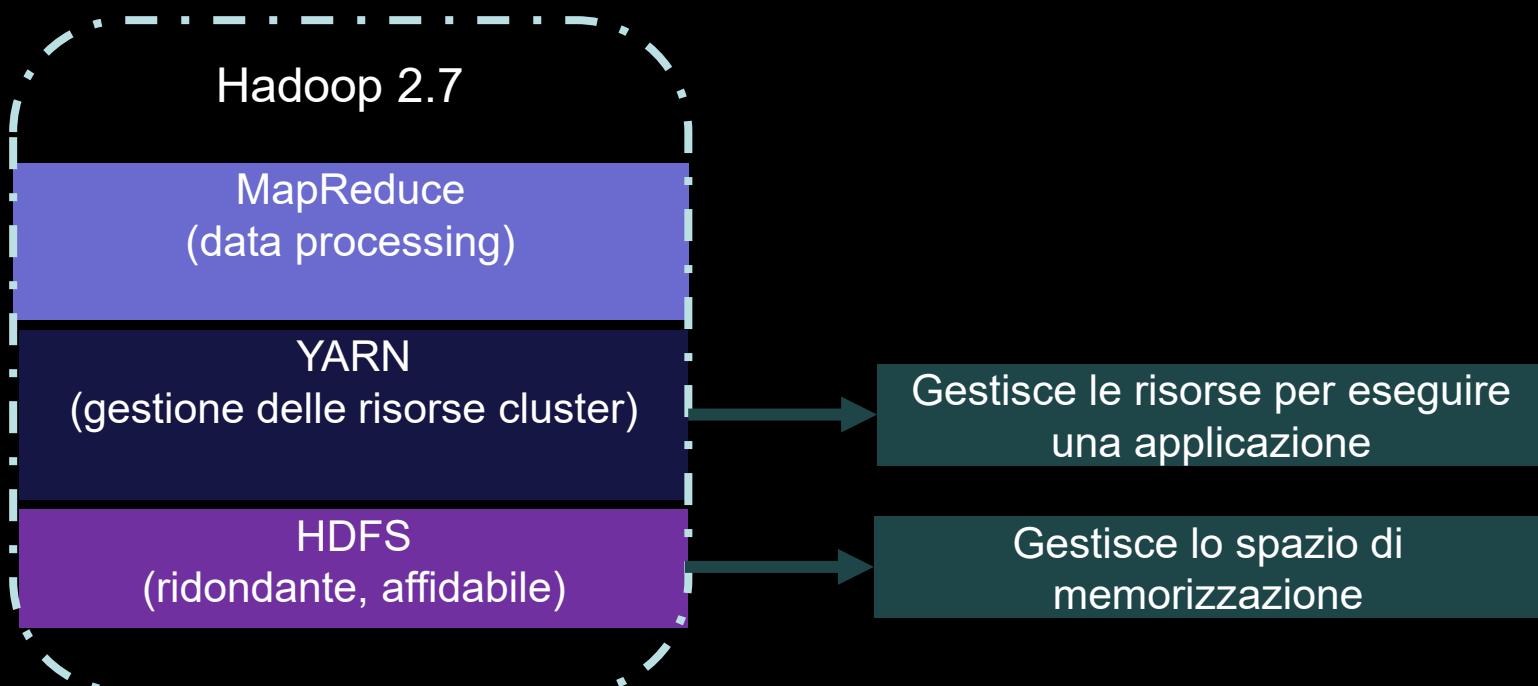
Infrastruttura di YARN

L'infrastruttura di YARN è responsabile di fornire le risorse computazionali per le applicazioni in esecuzione



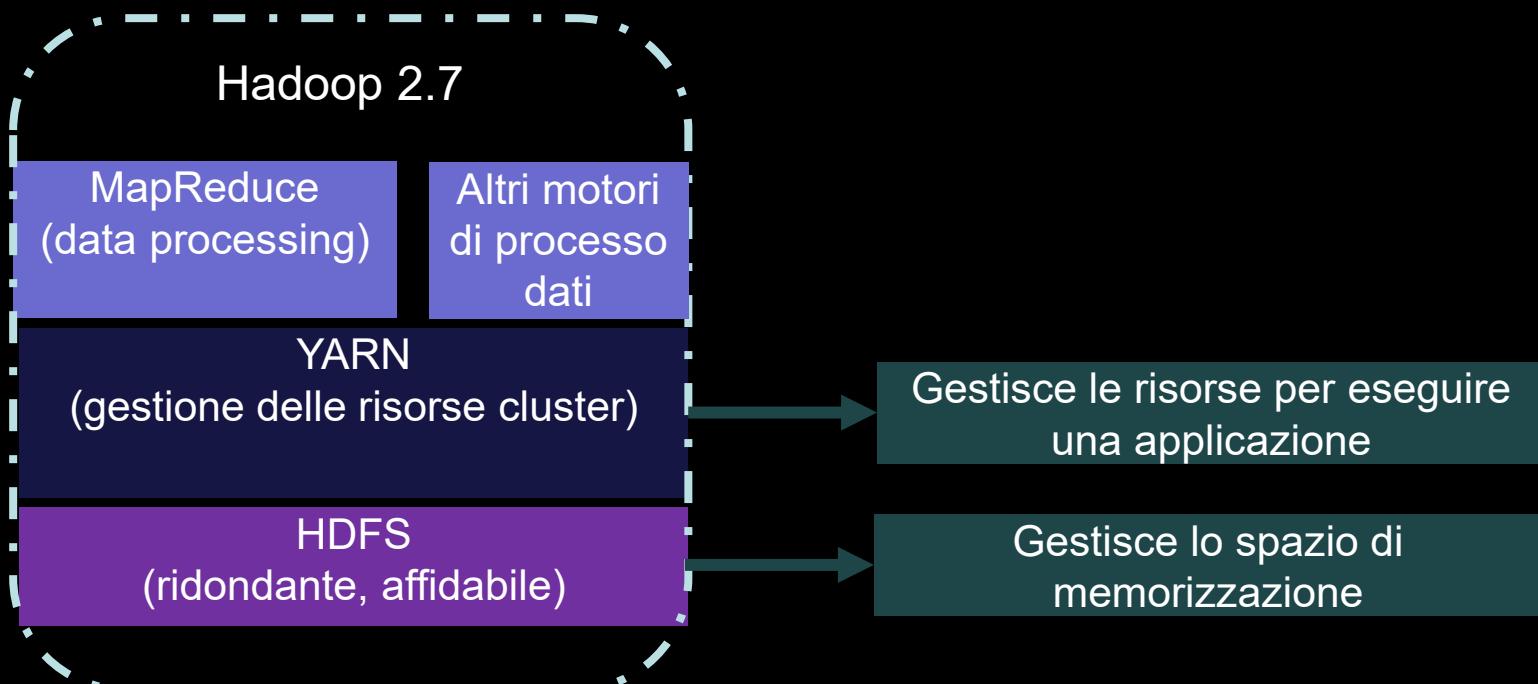
Infrastruttura di YARN

L'infrastruttura di YARN è responsabile di fornire le risorse computazionali per le applicazioni in esecuzione



Infrastruttura di YARN

L'infrastruttura di YARN è responsabile di fornire le risorse computazionali per le applicazioni in esecuzione



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master
- Node manager

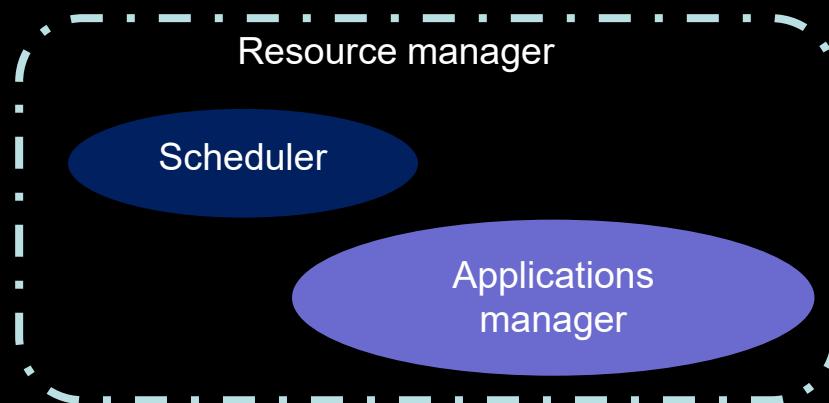
Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager



Risorse di processamento dati



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master



Risorse di processamento dati

Resource manager

Scheduler

Applications
manager

App master

App master

App master

Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master
- Node manager



Risorse di processamento dati

Resource manager

Scheduler

Applications
manager

Node manager

Container

App master

Data node

Node manager

Container

App master

Data node

Node manager

Container

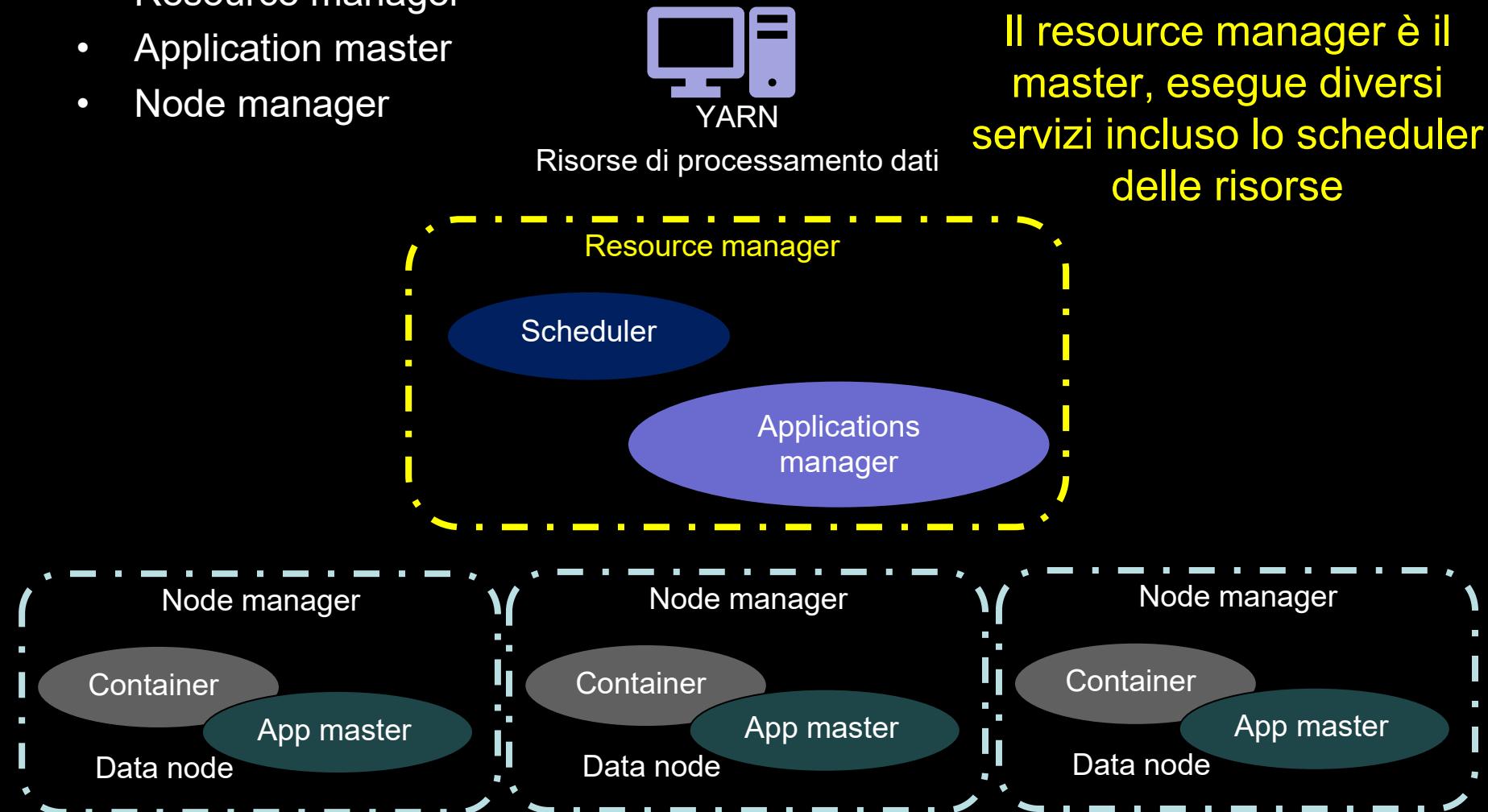
App master

Data node

Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

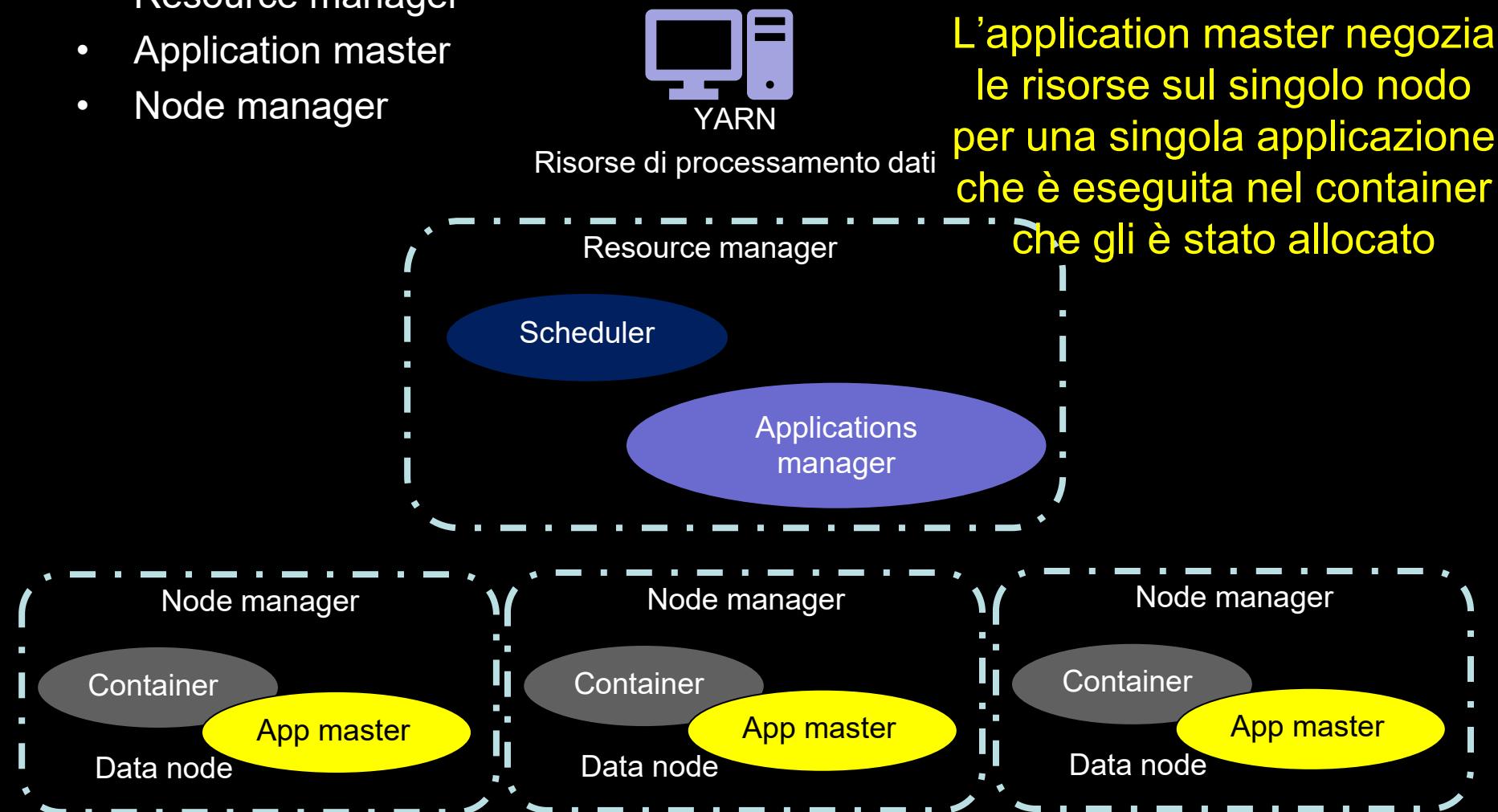
- Resource manager
- Application master
- Node manager



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

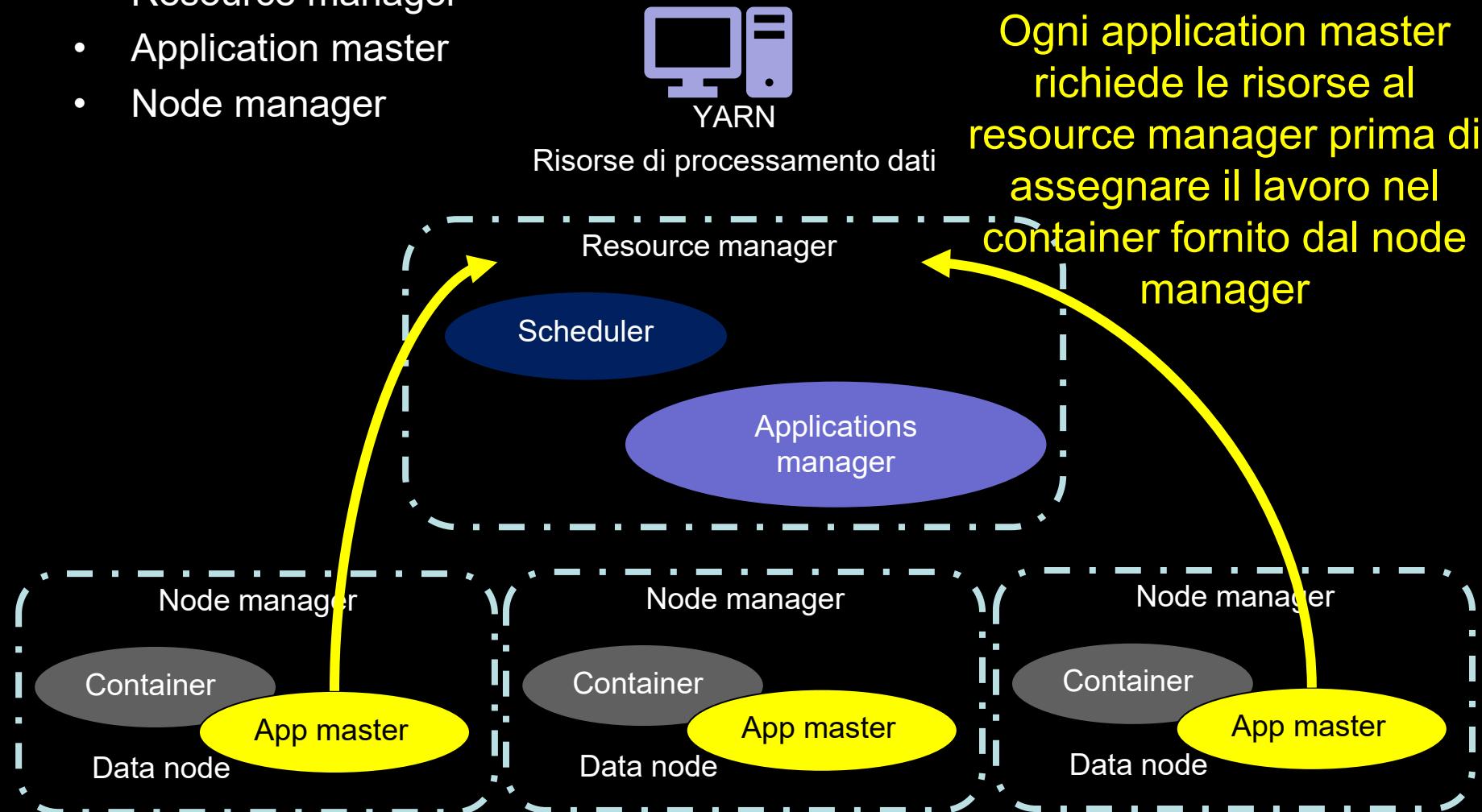
- Resource manager
- Application master
- Node manager



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master
- Node manager

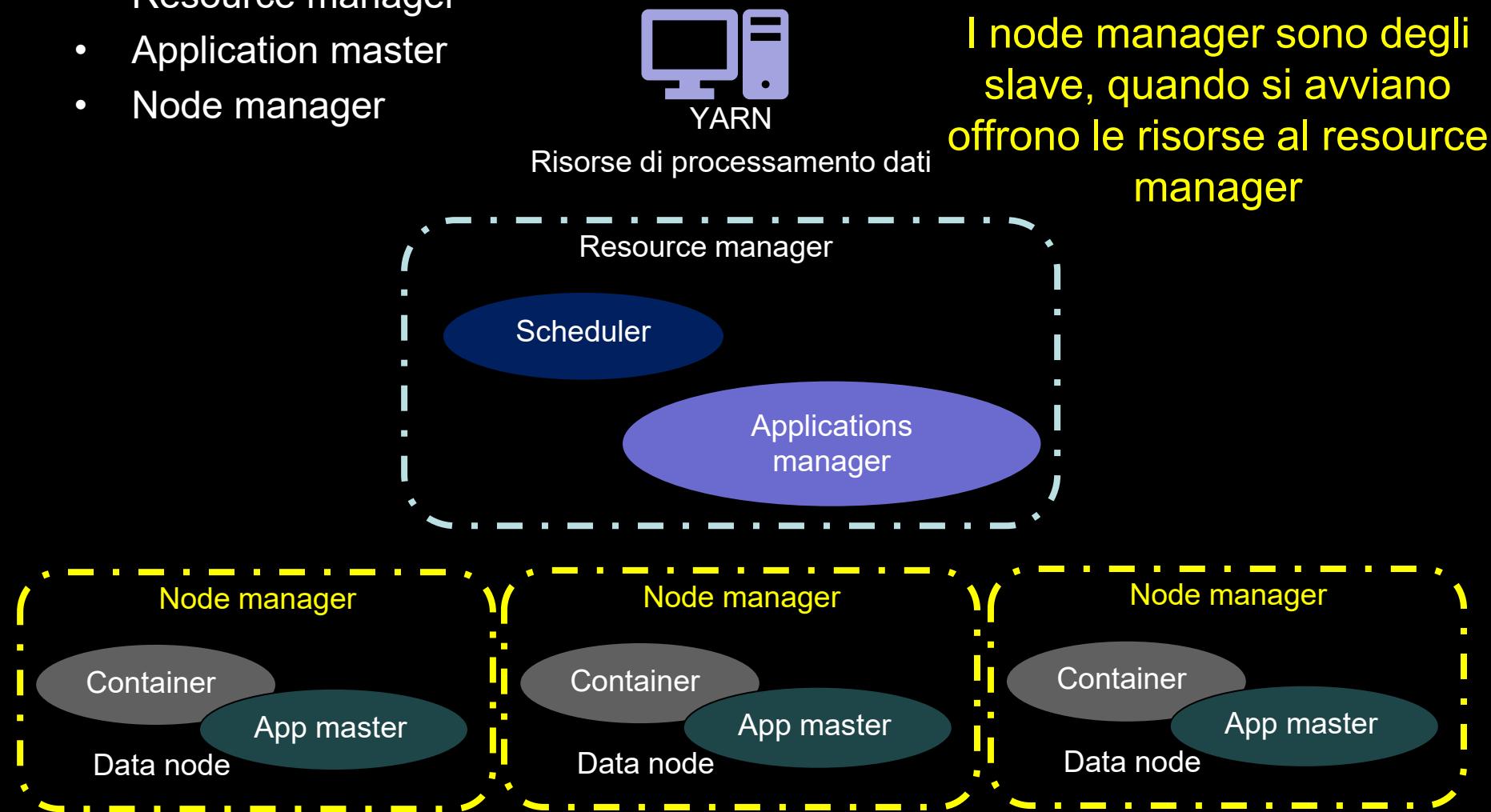


Ogni application master
richiede le risorse al
resource manager prima di
assegnare il lavoro nel
container fornito dal node
manager

Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

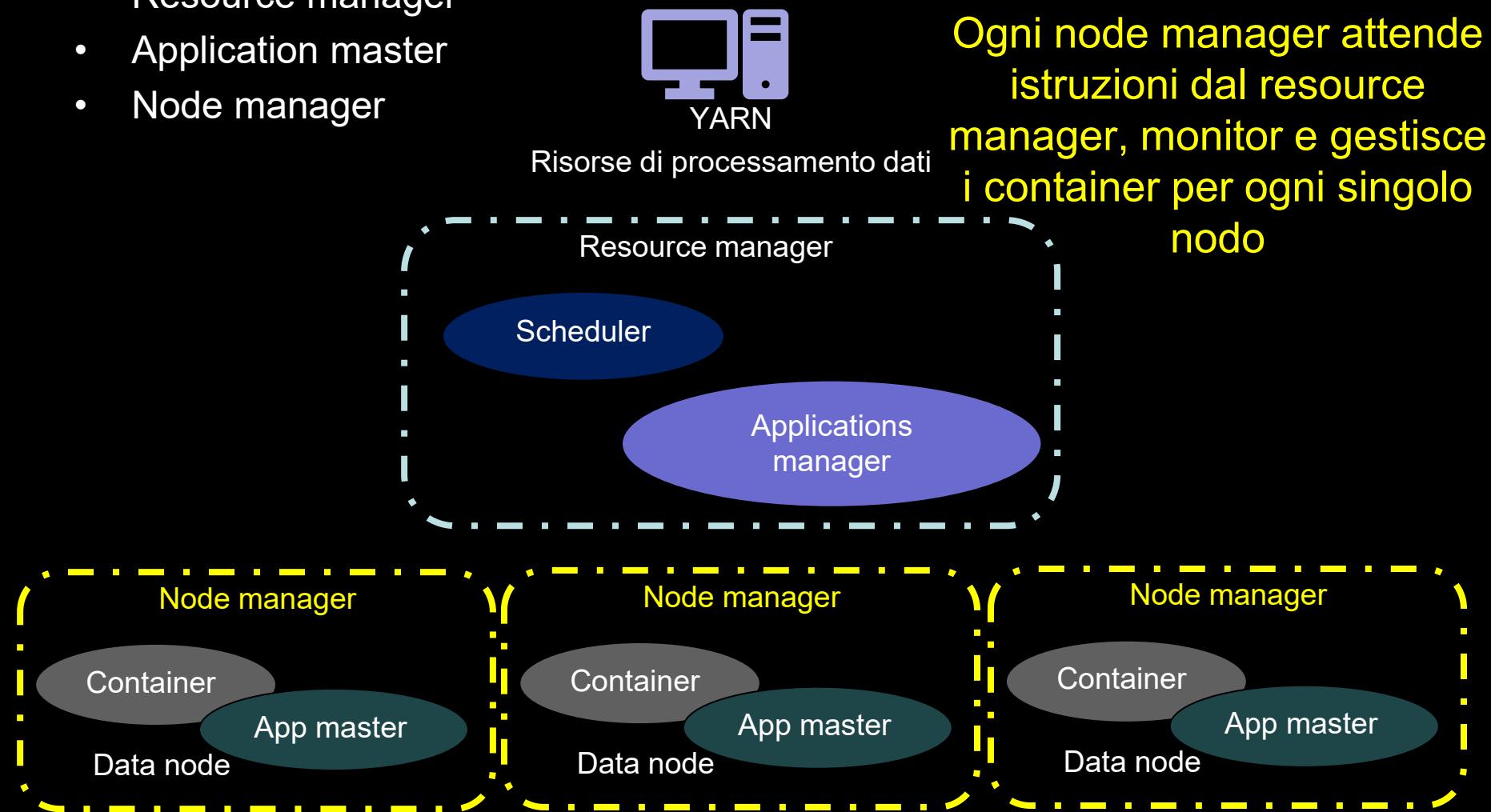
- Resource manager
- Application master
- Node manager



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

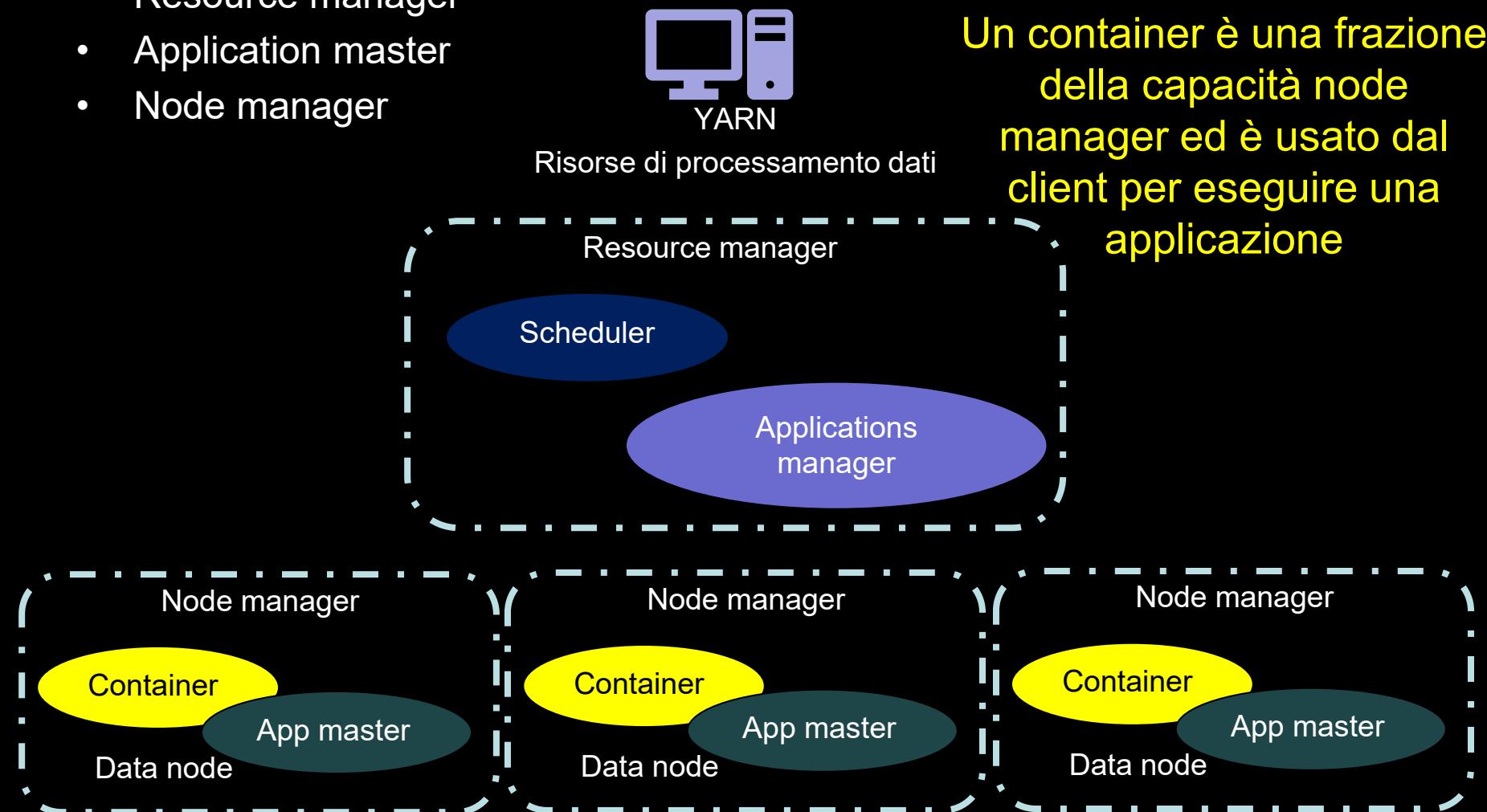
- Resource manager
- Application master
- Node manager



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master
- Node manager



Architettura di YARN

L'architettura di YARN è costituita da tre elementi fondamentali:

- Resource manager
- Application master
- Node manager



Risorse di processamento dati

Resource manager

Scheduler

Applications
manager

Node manager

Container

App master

Data node

Node manager

Container

App master

Data node

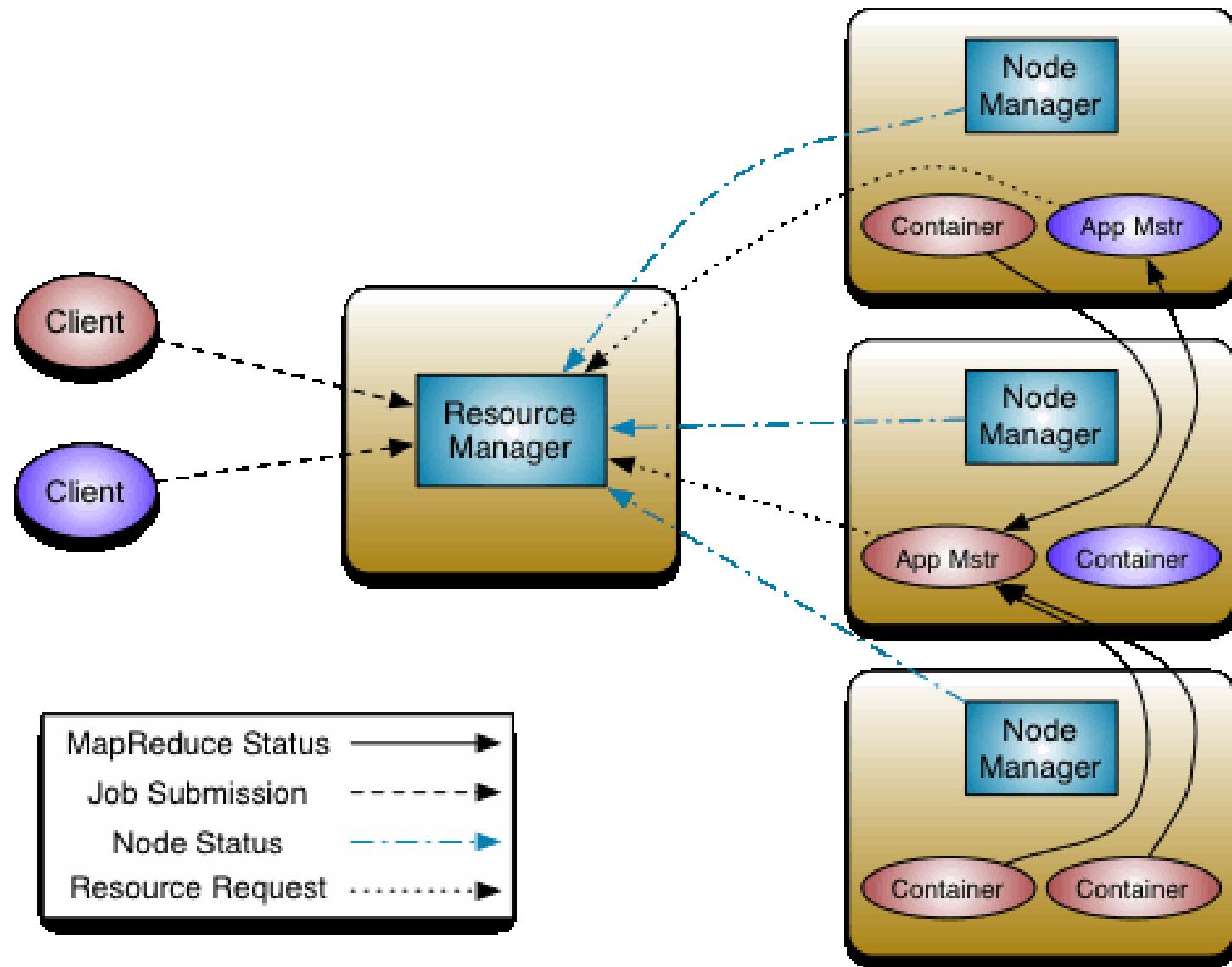
Node manager

Container

App master

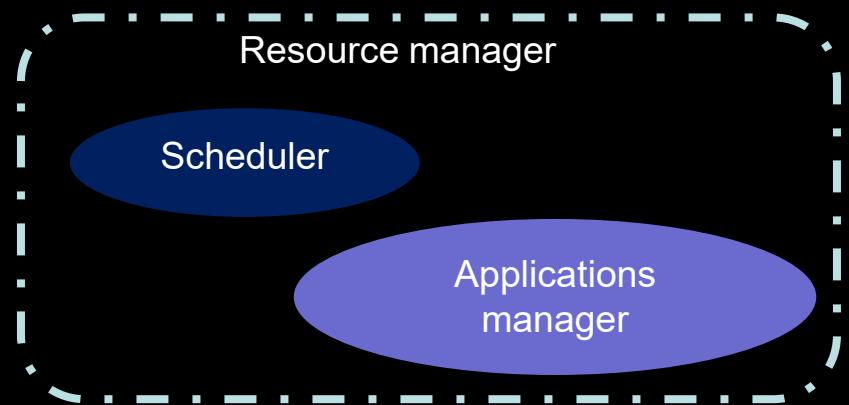
Data node

YARN – Documentazione ufficiale

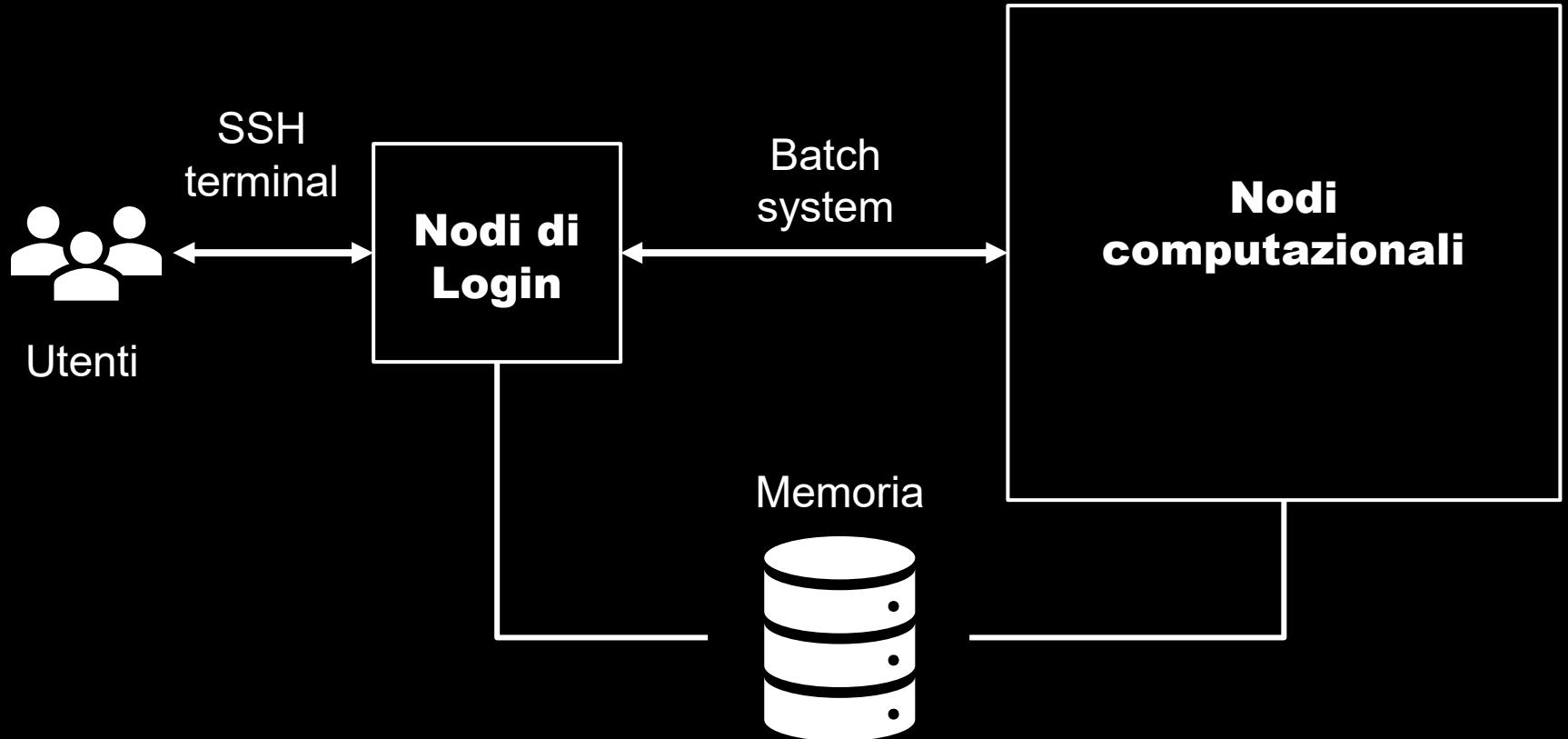


Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



HPC system layout

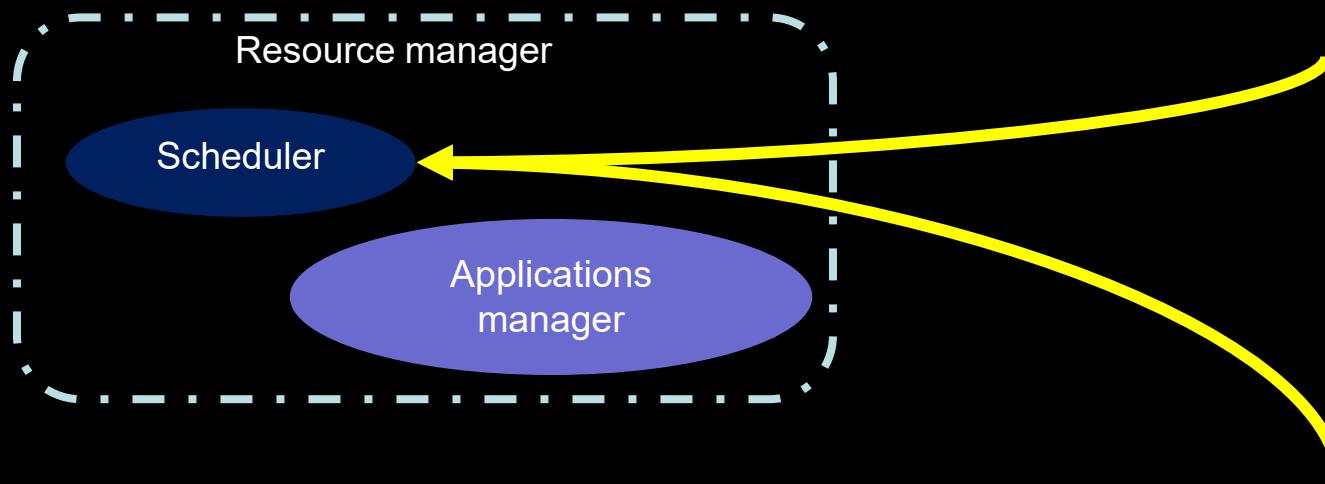


Gli utenti caricano/scaricano dati, e sottomettono il loro task computazionale tramite SSH terminal

Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware

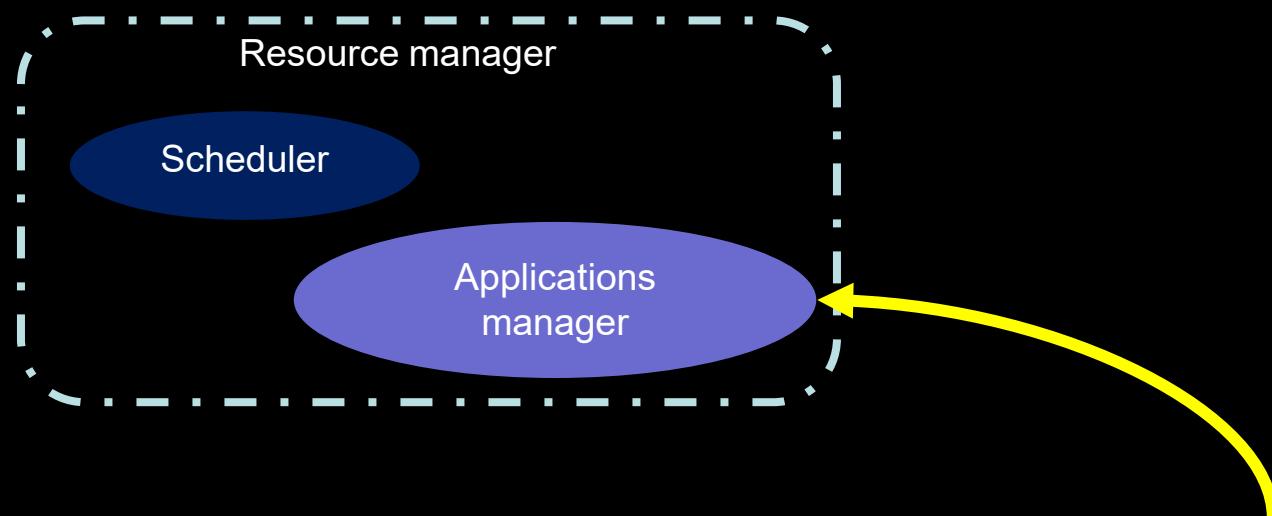
Non monitora lo stato delle applicazioni
Non le riavvia in caso di crash



Include un componente aggiuntivo
detto YarnScheduler, che permette
di cambiare tra diverse policy di
scheduling (es. fairScheduling,
capacityScheduling)

Architettura di YARN – Resource manager

L'application manager è una interfaccia che mantiene una lista dei processi che sono stati sottomessi, eseguiti in concorrenza o completati.



L'application manager è responsabile di accettare la sottomissione di jobs, negozia il primo container per eseguire l'applicazione ed esegue l'application master in caso di crash

Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware

ClientService

YARN scheduler

Resource manager

Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware

Resource manager

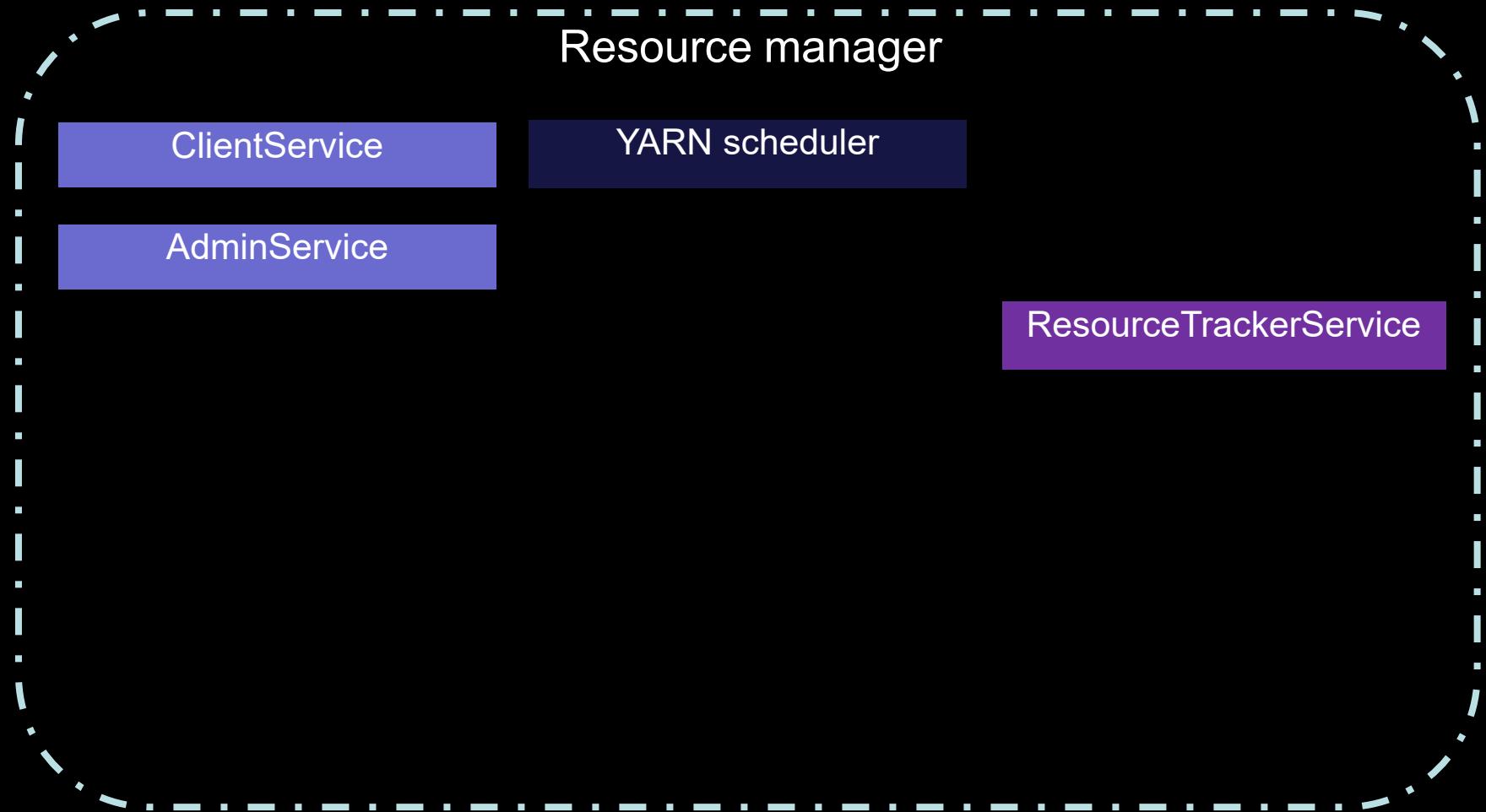
ClientService

YARN scheduler

AdminService

Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware

Resource manager

ClientService

YARN scheduler

NodeListManager

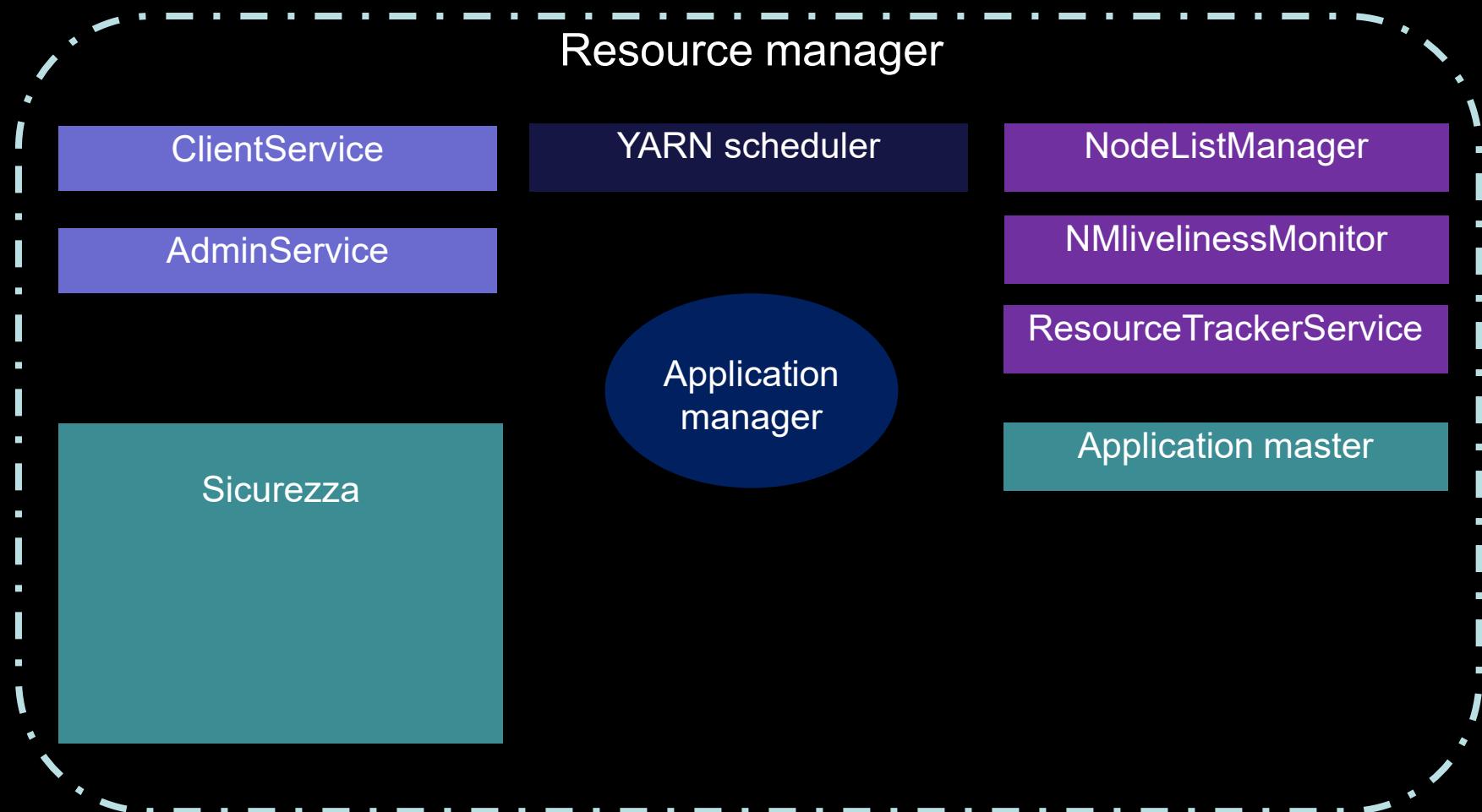
AdminService

NMlivelinessMonitor

ResourceTrackerService

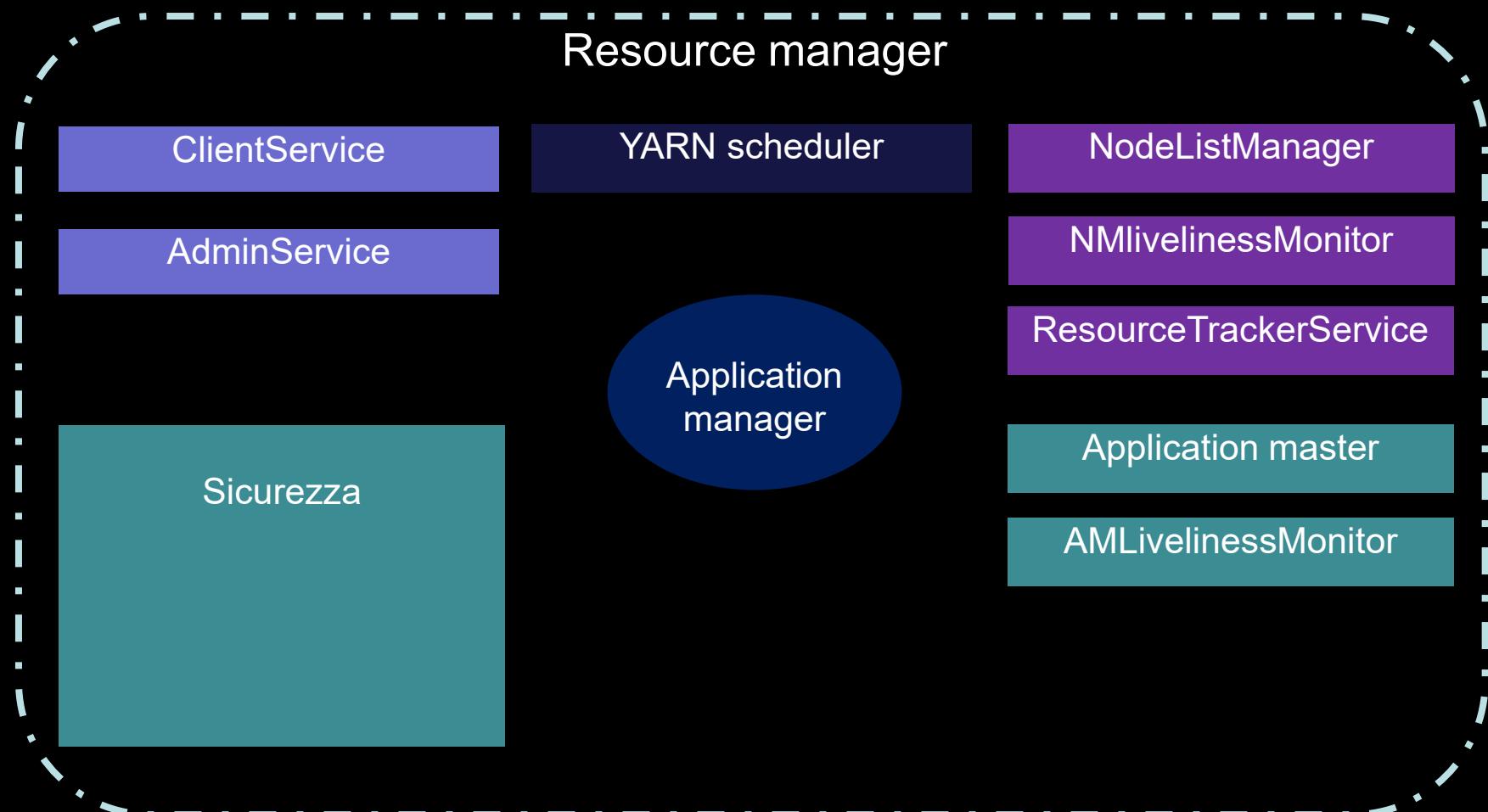
Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



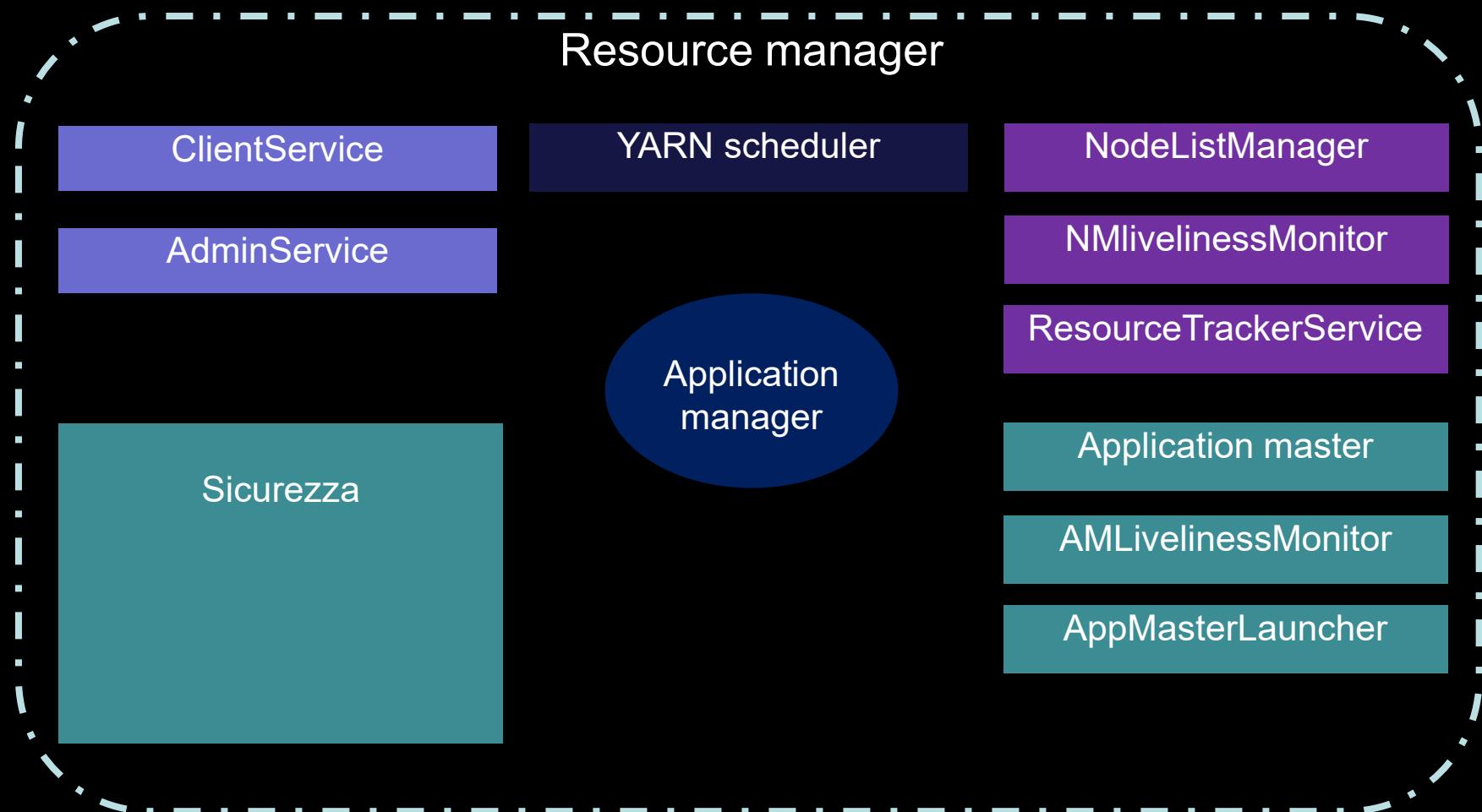
Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



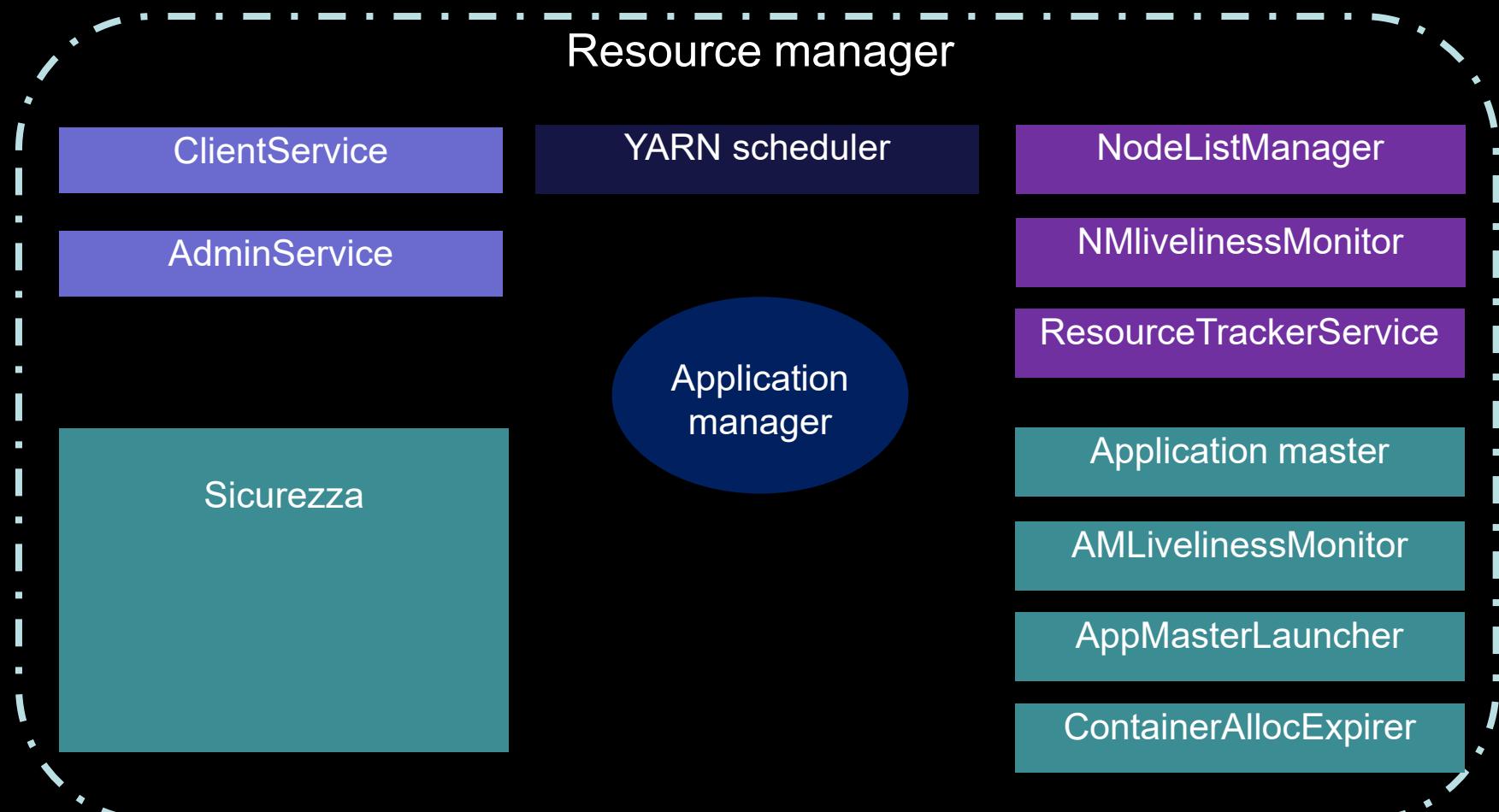
Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



Architettura di YARN – Resource manager

Il resource manager negozia le risorse disponibili nel cluster tra le applicazioni concorrenti – il fine è la massimizzazione dell'uso delle risorse hardware



Architettura di YARN – Resource manager

Rappresentava un unico punto critico

Questo problema è stato risolto successivamente con l'inserimento della caratteristica HA o high availability.

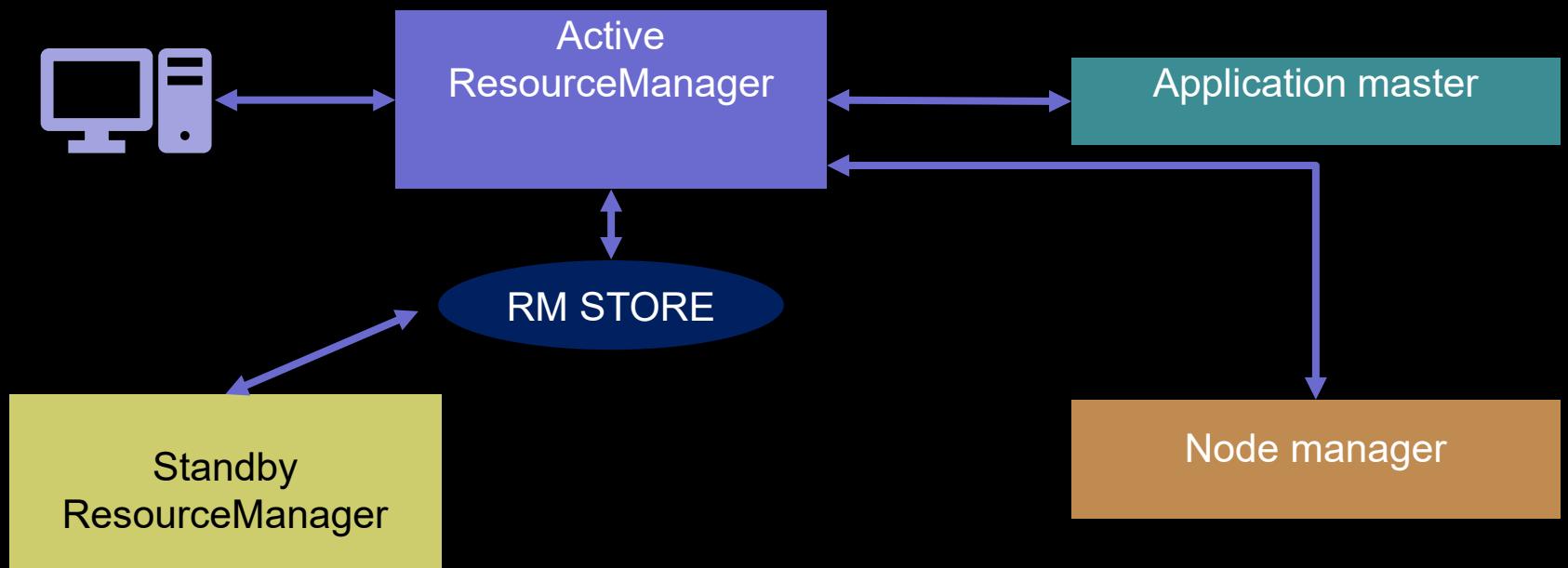
L'idea è sempre quella di far girare in parallelo una copia di resource managers attivo o in standby per rimuovere questo punto critico

Architettura di YARN – Resource manager

Rappresentava un unico punto critico

Questo problema è stato risolto successivamente con l'inserimento della caratteristica HA o high availability.

L'idea è sempre quella di far girare in parallelo una copia di resource managers attivo o in standby per rimuovere questo punto critico

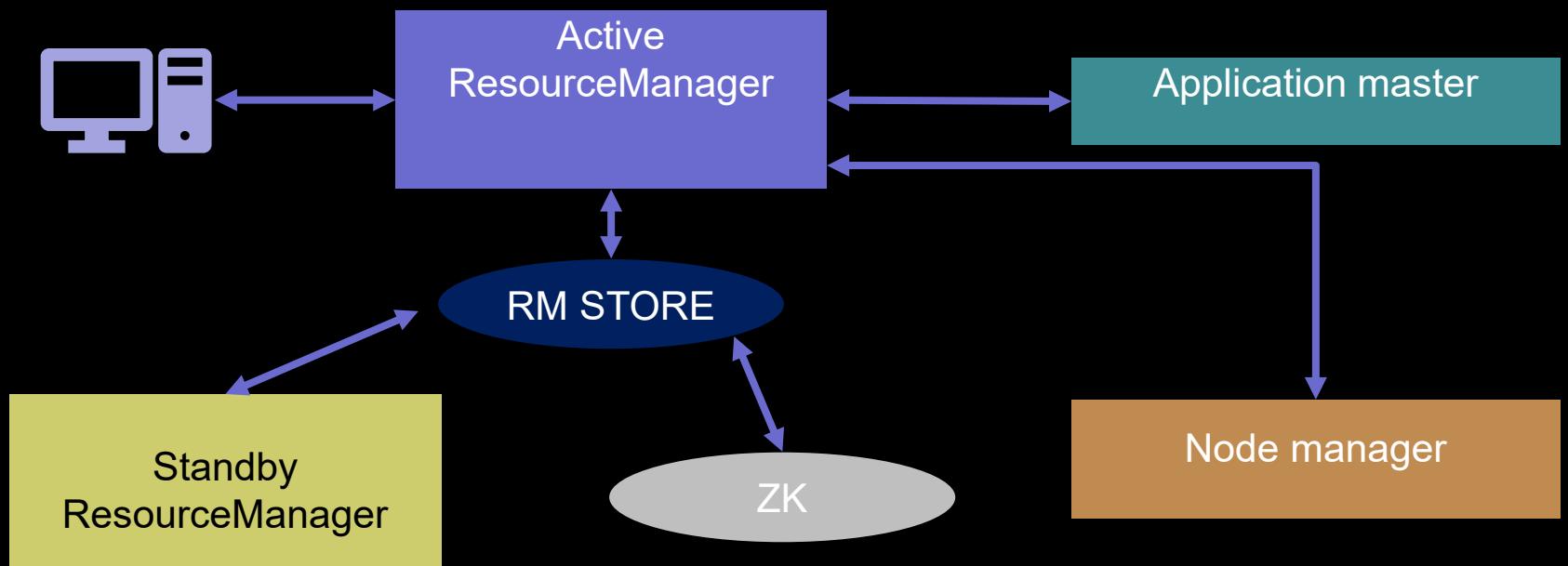


Architettura di YARN – Resource manager

Rappresentava un unico punto critico

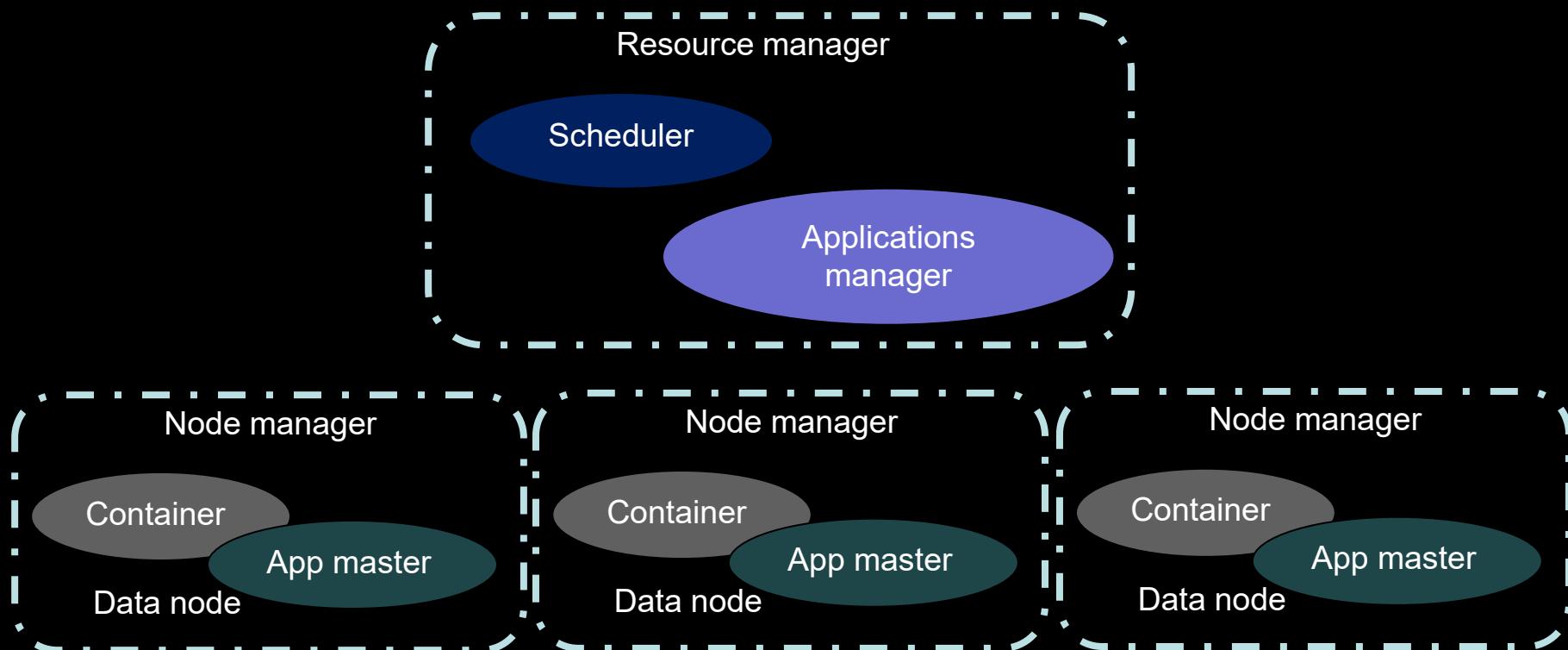
Questo problema è stato risolto successivamente con l'inserimento della caratteristica HA o high availability.

L'idea è sempre quella di far girare in parallelo una copia di resource managers attivo o in standby per rimuovere questo punto critico

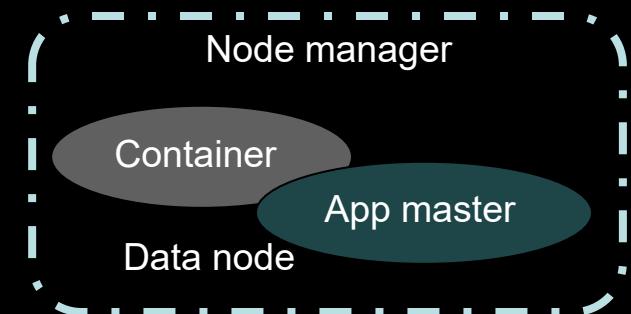


Architettura di YARN – Application master

L'application master in YARN è un componente dedicato che negozia risorse dal resource manager e lavora con il nodemanager per eseguire e monitorare il lavoro dei diversi container e il loro consumo di risorse



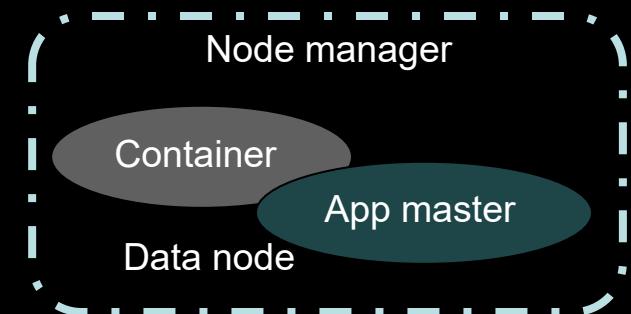
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni

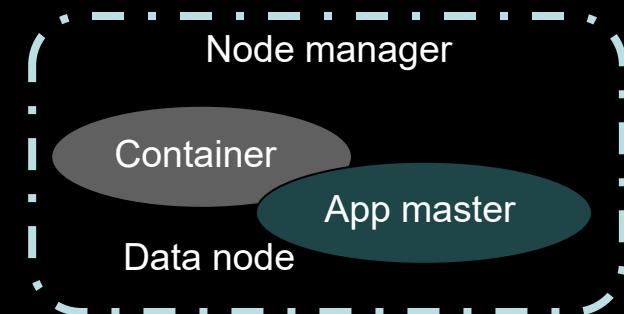
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse

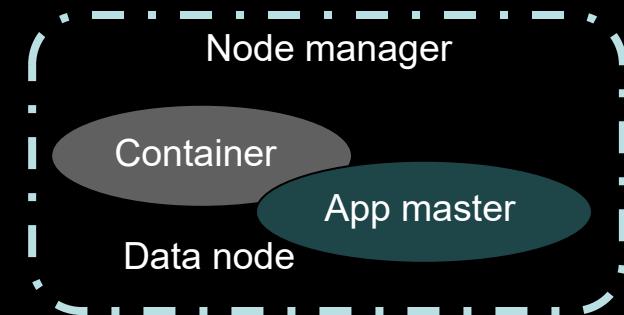
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione

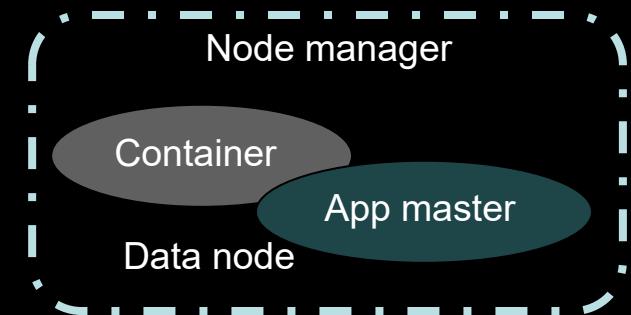
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults

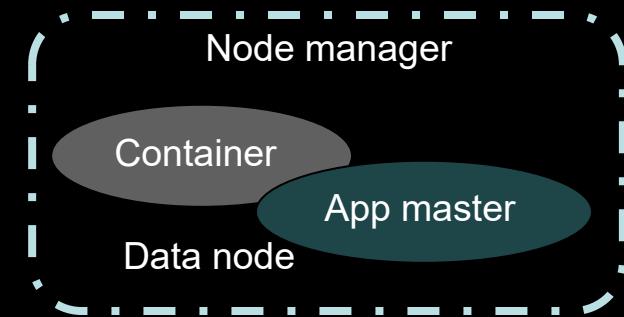
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults
- fornisce parametri di stato e altre metriche al resource manager

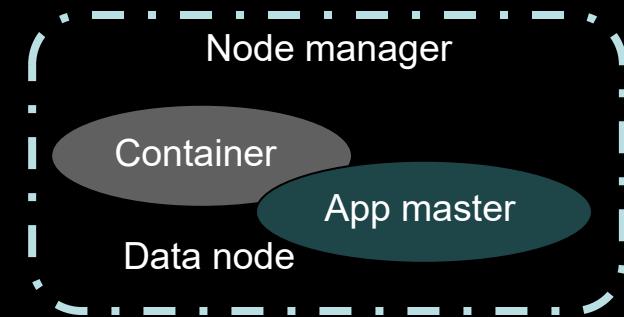
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults
- fornisce parametri di stato e altre metriche al resource manager
- può essere scritto in qualunque linguaggio

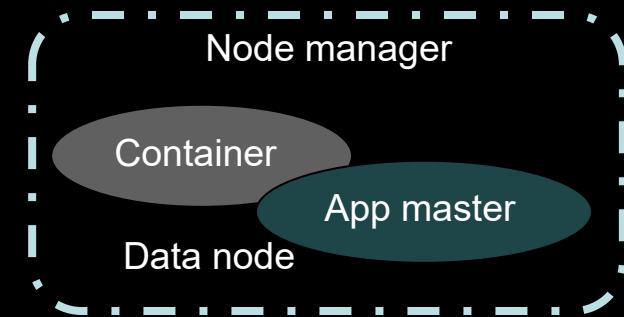
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults
- fornisce parametri di stato e altre metriche al resource manager
- può essere scritto in qualunque linguaggio
- usa la comunicazione di rete per interagire con il RM e il NM

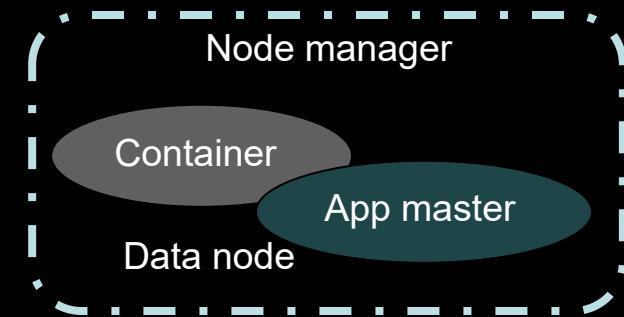
Architettura di YARN – Application master



L'application master

- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults
- fornisce parametri di stato e altre metriche al resource manager
- può essere scritto in qualunque linguaggio
- usa la comunicazione di rete per interagire con il RM e il NM
- non si esegue come un servizio “trusted”

Architettura di YARN – Application master



L'application master

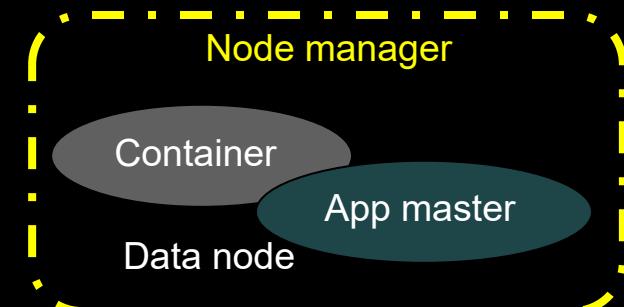
- Gestisce il lifecycle delle applicazioni
- fa aggiustamenti dinamici ai parametri di monitoraggio delle risorse
- gestisce il flusso di esecuzione
- gestisce eventuali faults
- fornisce parametri di stato e altre metriche al resource manager
- può essere scritto in qualunque linguaggio
- usa la comunicazione di rete per interagire con il RM e il NM
- non si esegue come un servizio “trusted”

ogni applicazione può avere la sua istanza di application master, ma si possono progettare application master per insiemi di applicazioni

Architettura di YARN – Node Manager

Quando un contenitore è inserito nella lista per una applicazione, il node manager imposta il l'ambiente del contenitore, includendo vincoli di risorse specificati nella richiesta e ogni dipendenza necessaria

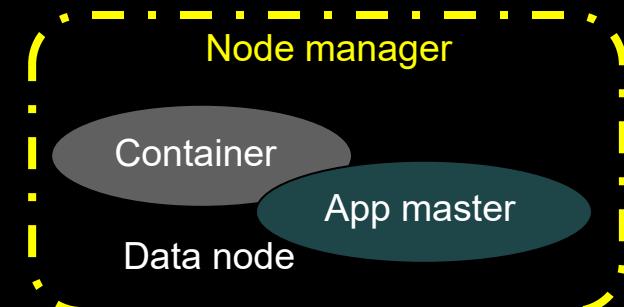
Architettura di YARN – Node Manager



Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management

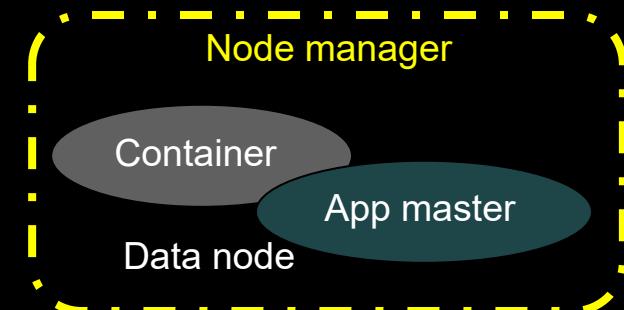
Architettura di YARN – Node Manager



Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management
- Dipendenze

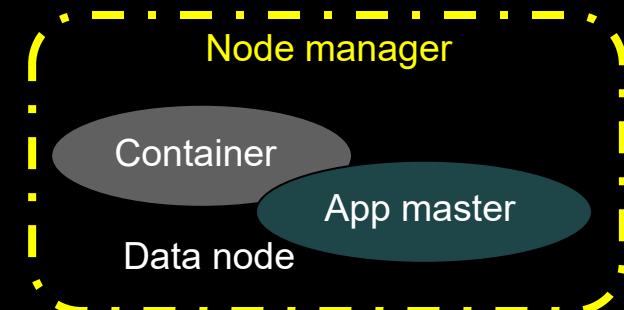
Architettura di YARN – Node Manager



Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management
- Dipendenze
- Utilizzo delle risorse nel nodo e nei container

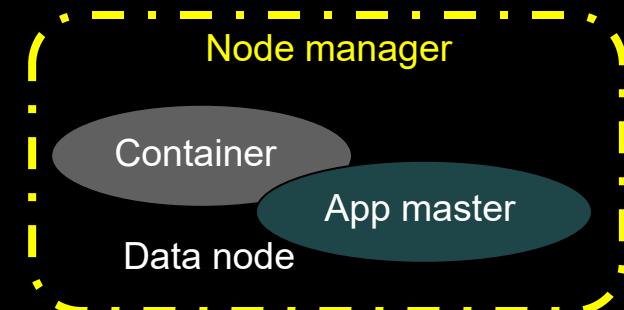
Architettura di YARN – Node Manager



Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management
- Dipendenze
- Utilizzo delle risorse nel nodo e nei container
- Stato di operatività del nodo (heartbeat)

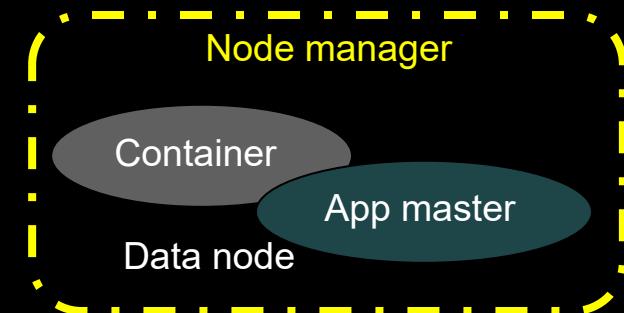
Architettura di YARN – Node Manager



Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management
- Dipendenze
- Utilizzo delle risorse nel nodo e nei container
- Stato di operatività del nodo (heartbeat)
- Gestisce i file di log

Architettura di YARN – Node Manager



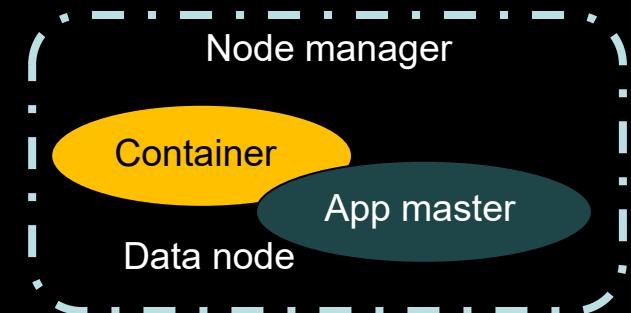
Il node manager è eseguito su ogni nodo e gestisce:

- Container lifecycle management
- Dipendenze
- Utilizzo delle risorse nel nodo e nei container
- Stato di operatività del nodo (heartbeat)
- Gestisce i file di log
- Riporta lo stato dei nodi e dei container al resource manager

Architettura di YARN – Container

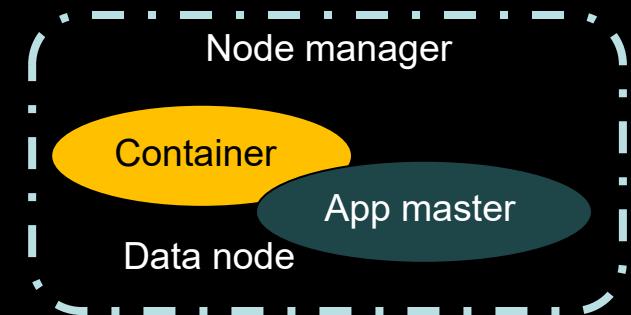
Un container è il risultato di una allocazione delle risorse corretta, ciò significa che il resource manager ha assegnato delle risorse specifiche in uno specifico nodo ad una specifica applicazione

Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

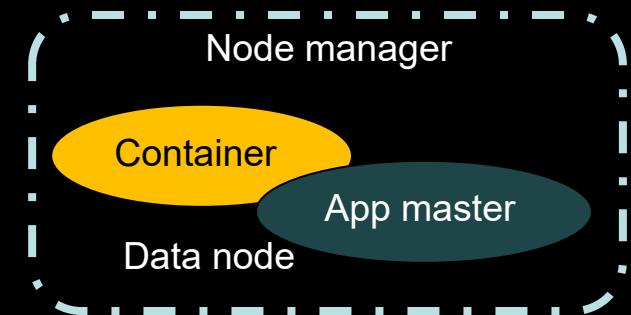
Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

- Variabili di ambiente

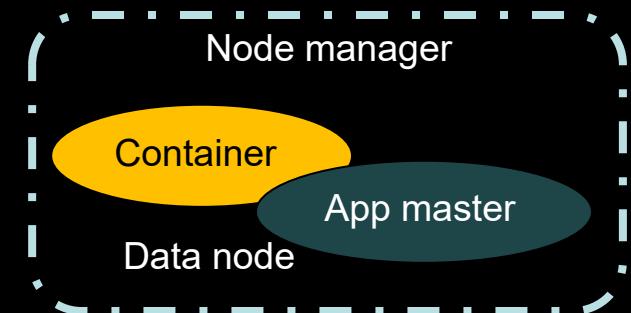
Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

- Variabili di ambiente
- Dipendenze

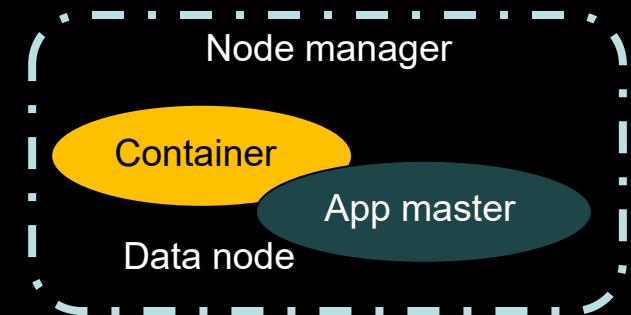
Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

- Variabili di ambiente
- Dipendenze
- Risorse locali inclusi file e dati condivisi

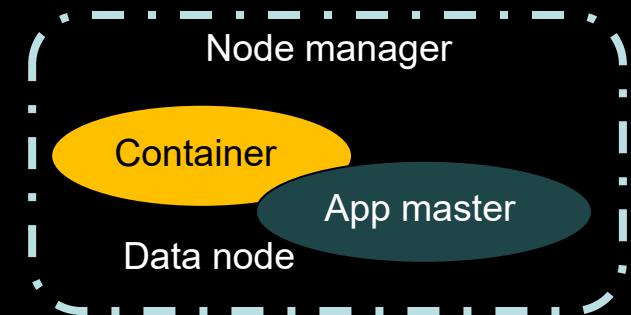
Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

- Variabili di ambiente
- Dipendenze
- Risorse locali inclusi file e dati condivisi
- Token di sicurezza (RPC)

Architettura di YARN – Container



Per lanciare un container, l'application master deve fornire un container launch context (CLC) che include le seguenti informazioni:

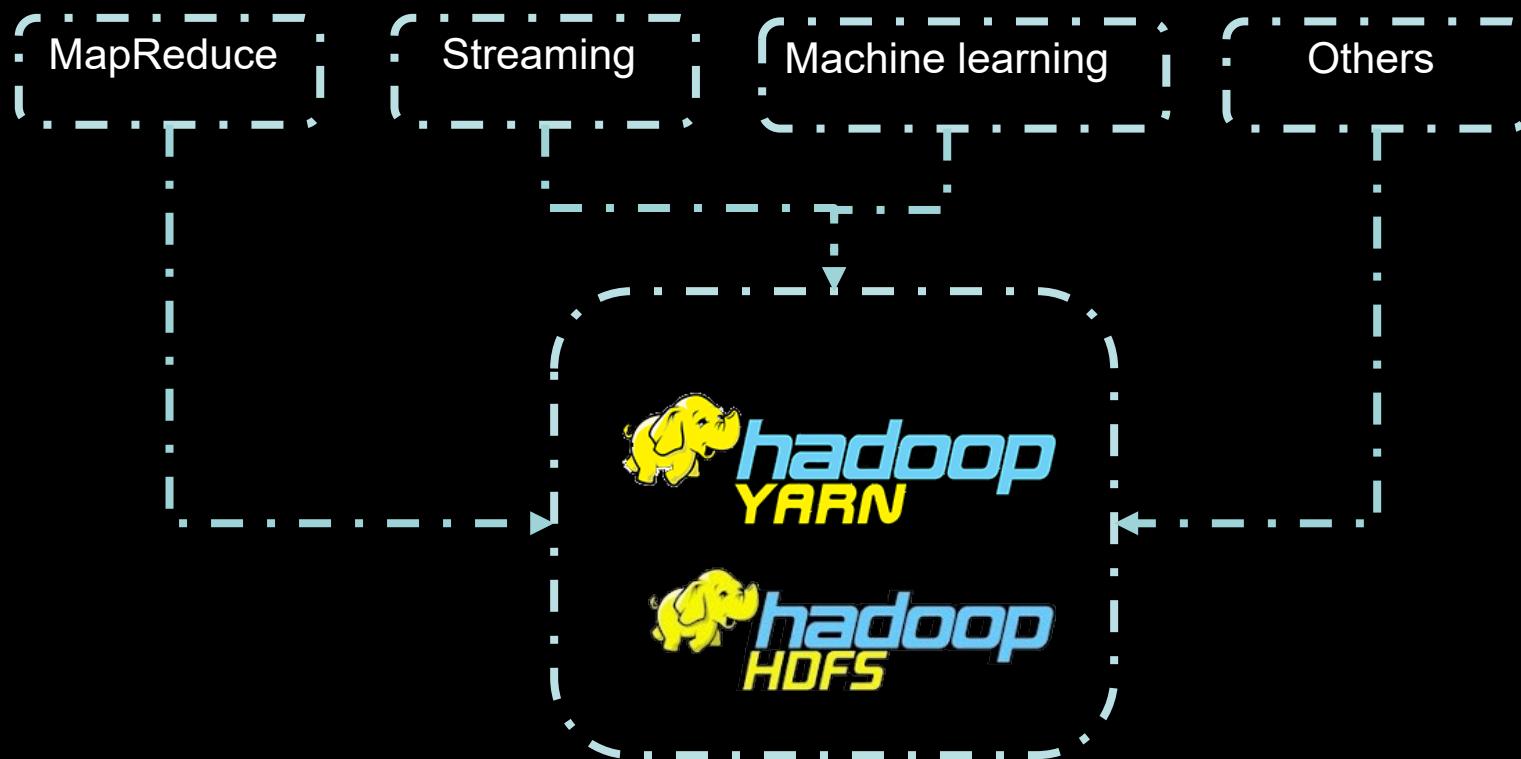
- Variabili di ambiente
- Dipendenze
- Risorse locali inclusi file e dati condivisi
- Token di sicurezza (RPC)
- I comandi necessary per creare il processo di applicazione che si intende lanciare

Applicazioni su YARN

Ci possono essere molte diversi carichi computazionali che sono eseguiti su un cluster Hadoop

Applicazioni su YARN

Ci possono essere molte diverse carichi computazionali che sono eseguiti su un cluster Hadoop



Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per una lanciare una applicazione

Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per lanciare una applicazione

1. Il client sottomette una applicazione al resource manager

Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per una lanciare una applicazione

1. Il client sottomette una applicazione al resource manager
2. Il resource manager alloca un container per eseguire l'applicazione

Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per lanciare una applicazione

1. Il client sottomette una applicazione al resource manager
2. Il resource manager alloca un container per eseguire l'applicazione
3. L'application master contatta il node manager per allocare il container

Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per lanciare una applicazione

1. Il client sottomette una applicazione al resource manager
2. Il resource manager alloca un container per eseguire l'applicazione
3. L'application master contatta il node manager per allocare il container
4. Il corrispettivo node manager lancia il container

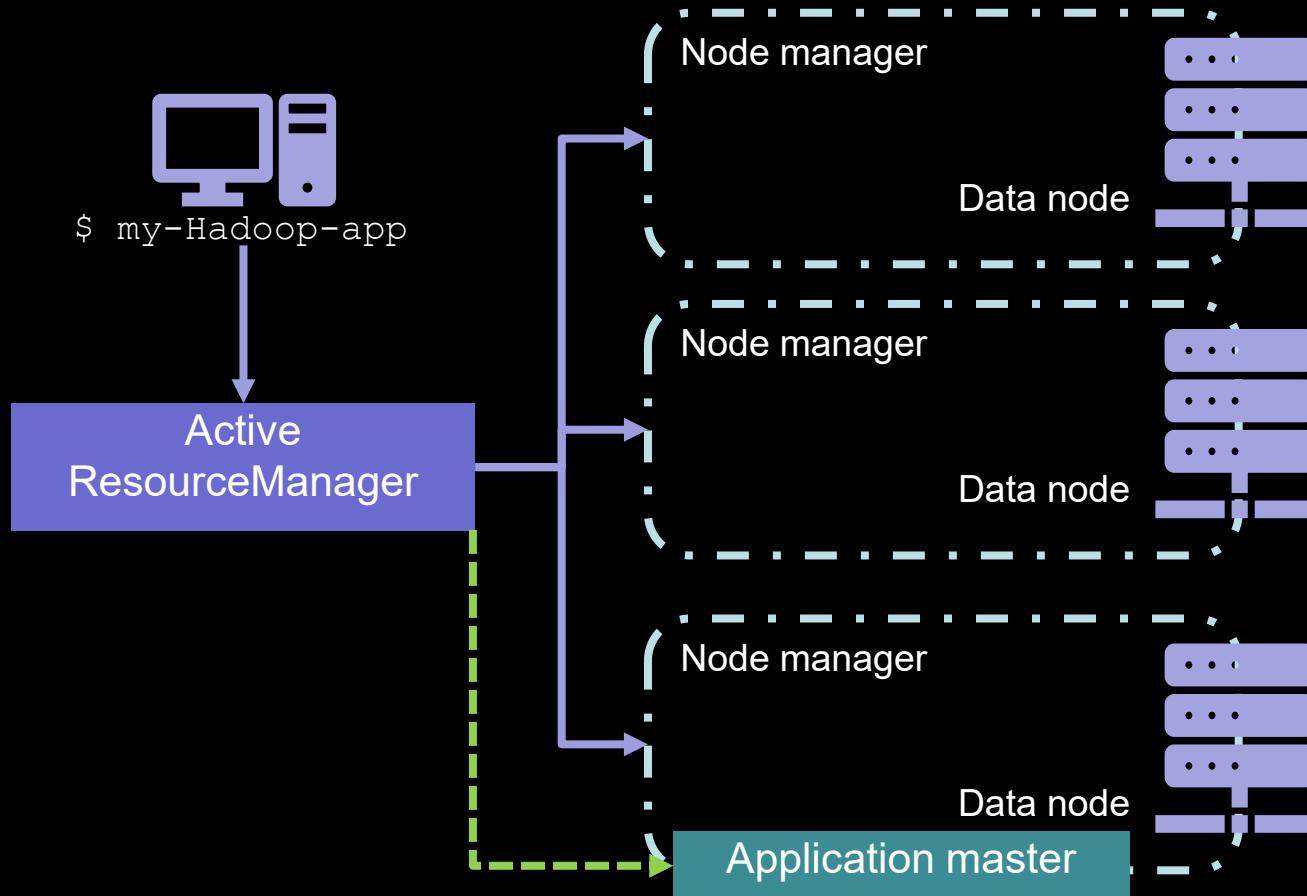
Passi di esecuzione di una applicazione in YARN

In generale ci sono 5 passi fondamentali per lanciare una applicazione

1. Il client sottomette una applicazione al resource manager
2. Il resource manager alloca un container per eseguire l'applicazione
3. L'application master contatta il node manager per allocare il container
4. Il corrispettivo node manager lancia il container
5. Il container esegue l'application master

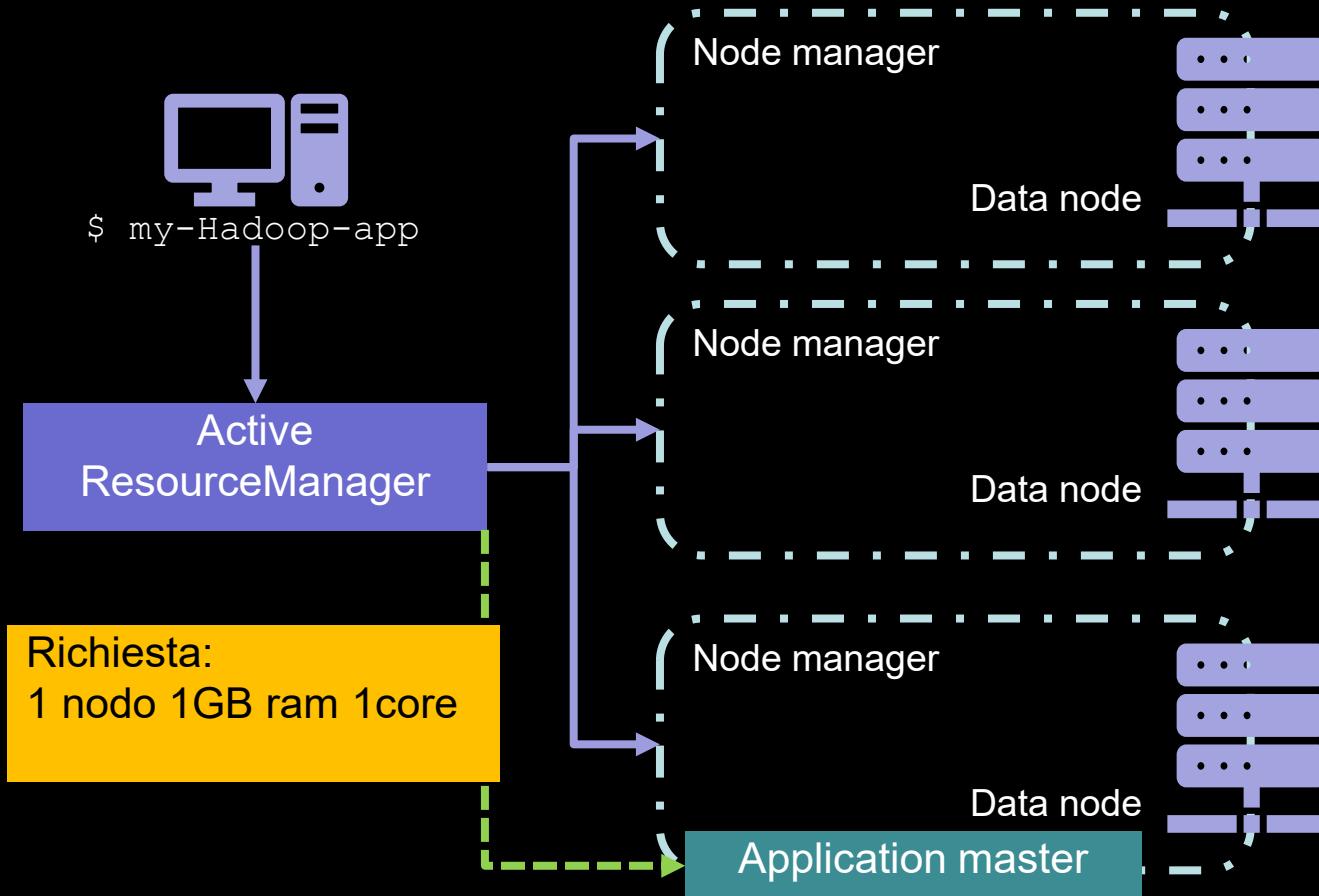
Step 1 – sottomissione della applicazione al RM

Gli utenti sottomettono i lavori al resource manager tramite il comando Hadoop jar



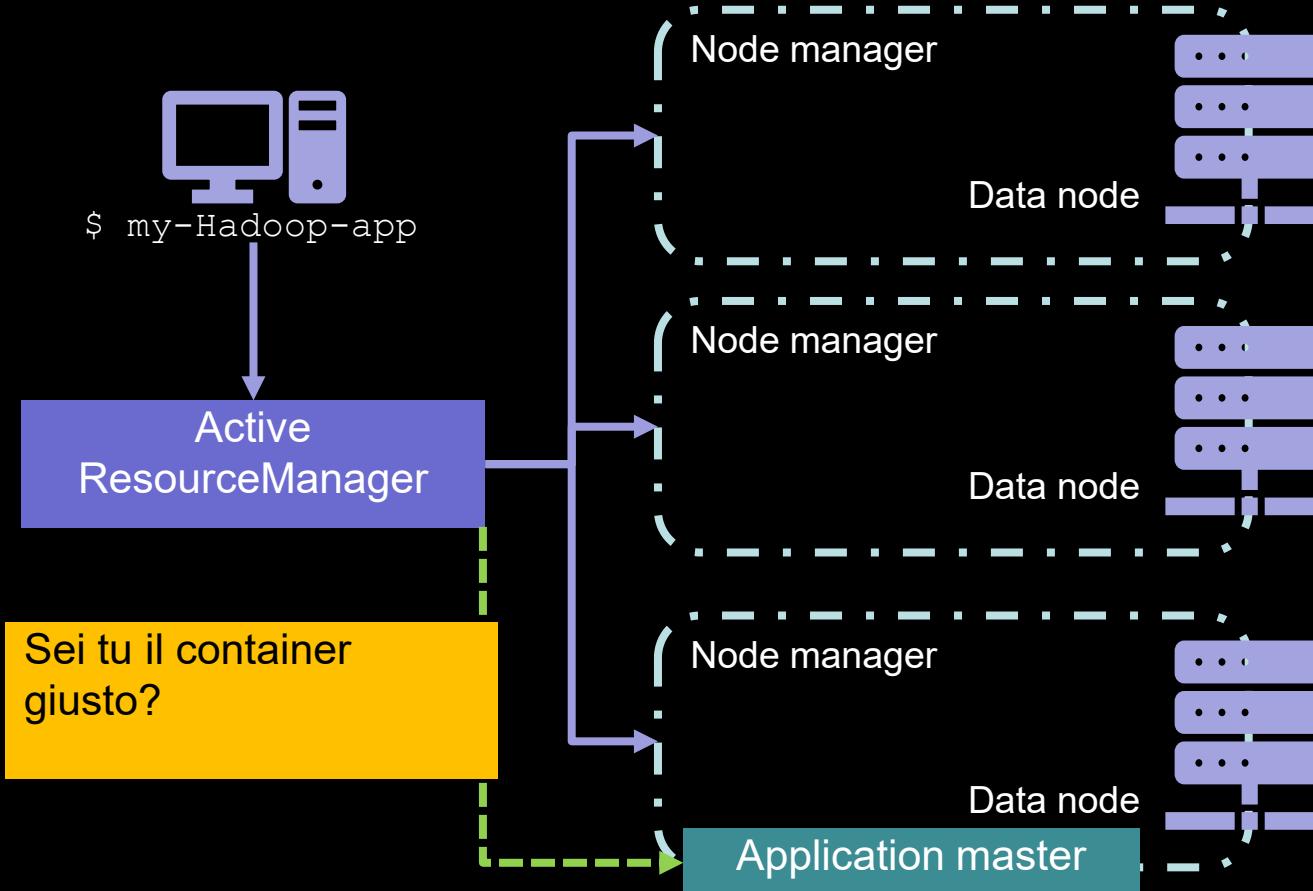
Step 2 – Allocazione risorse nel container

Quando il resource manager accetta la nuova applicazione, il primo passo per lo scheduler è selezionare un container



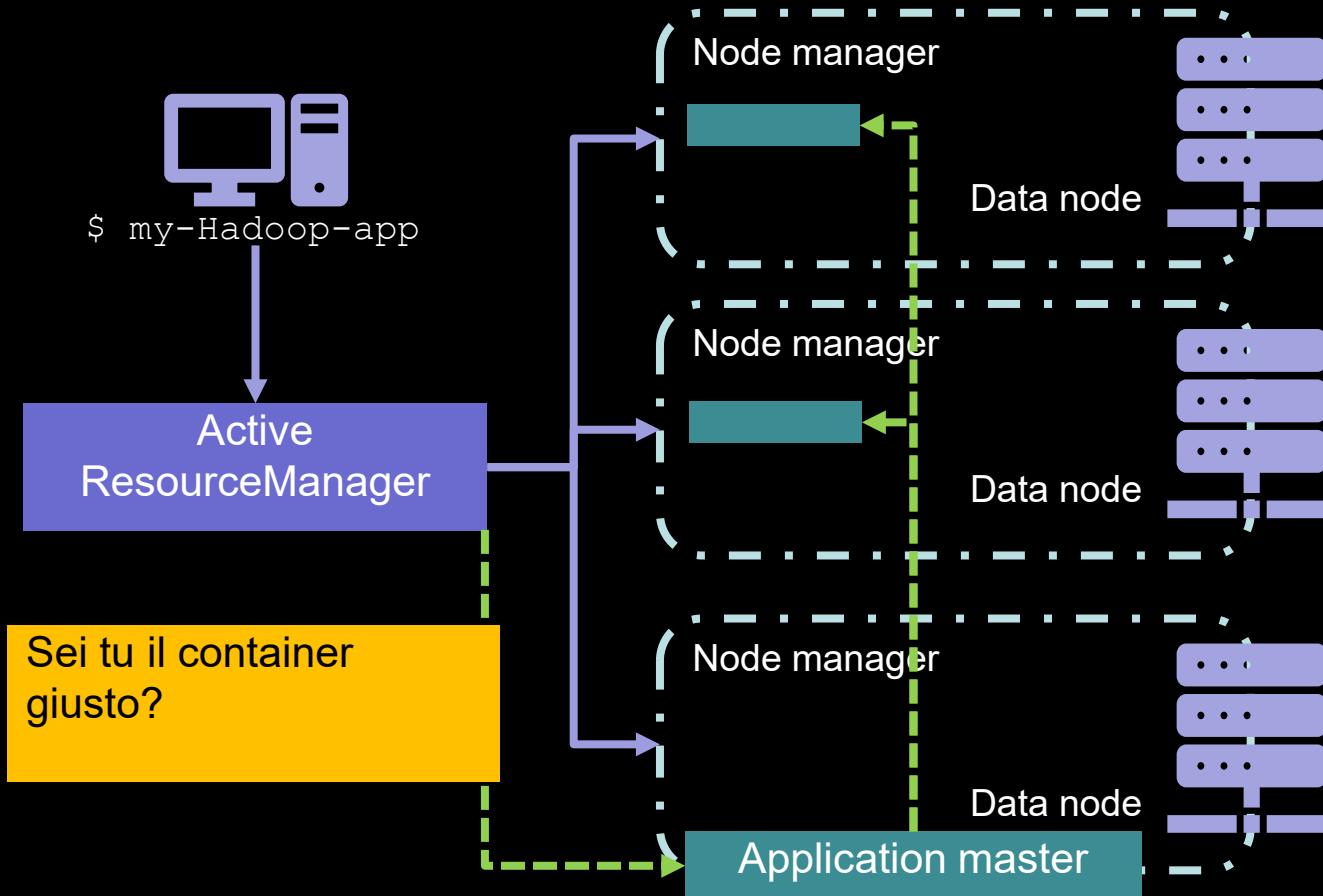
Step 3 – L'application master comunica con il node manager

Quando il container è allocato, l'application master chiede al nodeManager quale container è stato allocato per usare le risorse richieste



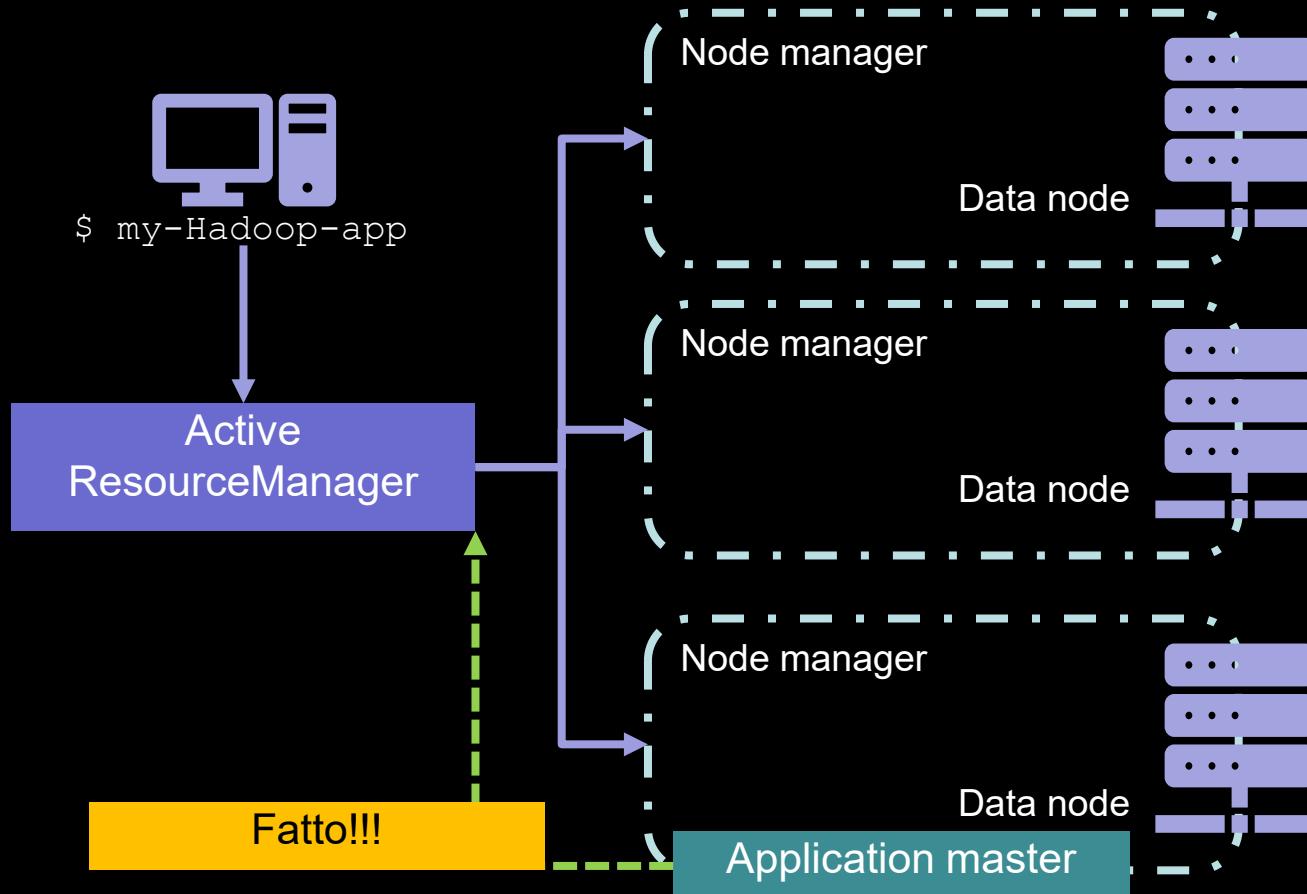
Step 4 – Il resource manager lancia il container

Il node manager non monitora i task, monitora solo le risorse nel container



Step 5 – I container esegue l'applicazione

Quando l'applicazione è completata, l'application master si termina e rilascia il contenitore/i allocato





Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

BIG DATA e Analisi dei Dati

Lezione 2.3 – Progettazione cluster hadoop

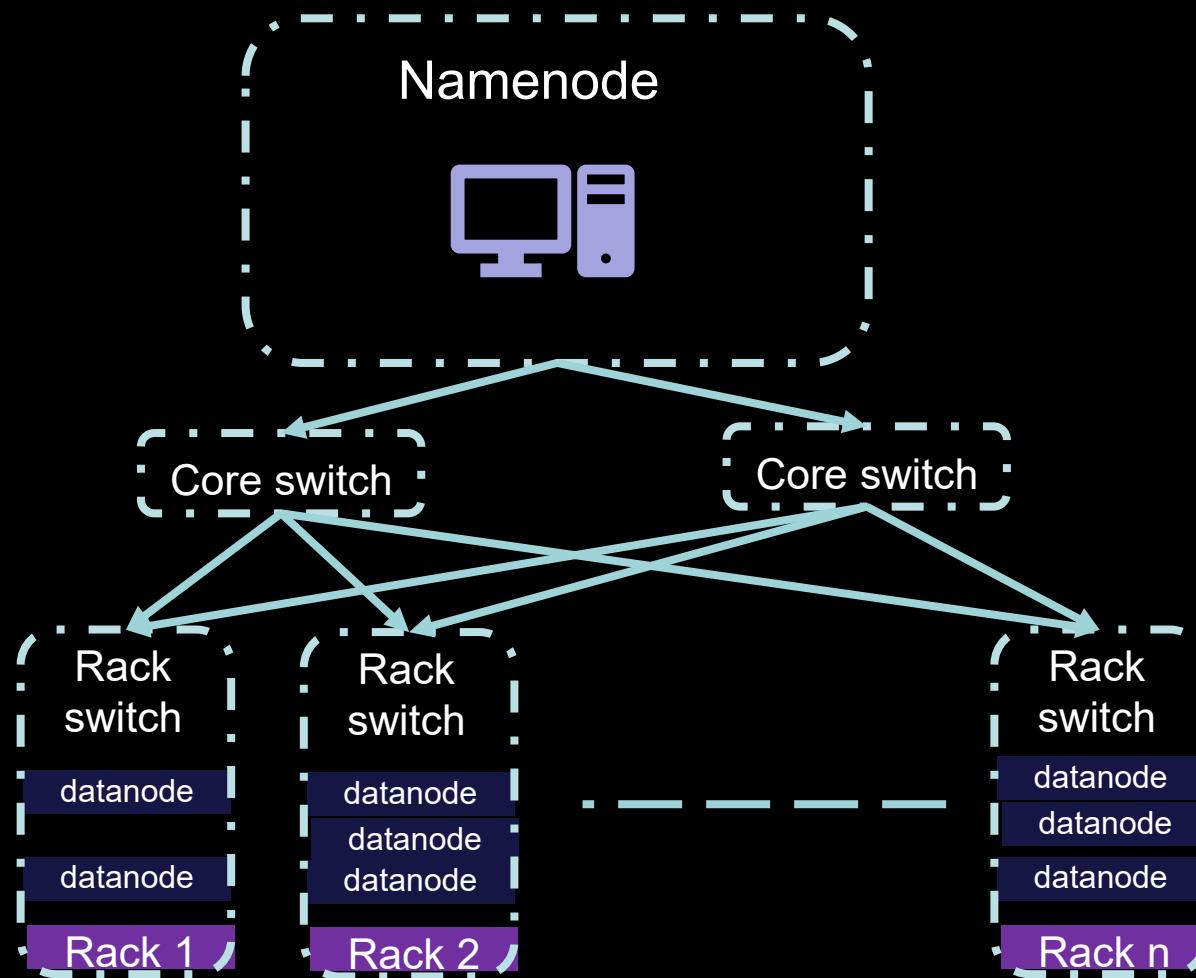
Mauro Bellone,
Robotics and AI researcher

bellonemauro@gmail.com
www.maurobellone.com

Hadoop cluster elementi di progettazione

Hadoop cluster elementi di progettazione

- Selezione hardware – **Namenode, datanodes e rete**



Hadoop cluster elementi di progettazione

- **Selezione hardware**

Configurazione media (all around, deep storage, 1 Gb Ethernet)

CPU	2 × 6 core 2.9 Ghz/15 MB cache
Memory	64 GB DDR3-1600 ECC
Disk controller	SAS 6 Gb/s
Disks	12 × 3 TB LFF SATA II 7200 RPM
Network controller	2 × 1 Gb Ethernet
Note	CPU con Intel's Hyper-Threading e QPI.

Hadoop cluster elementi di progettazione

- **Selezione hardware**

Configurazione alta

CPU	2 × 6 core 2.9 Ghz/15 MB cache
Memory	96 GB DDR3-1600 ECC
Disk controller	2 × SAS 6 Gb/s
Disks	24 × 1 TB SFF Nearline/MDL SAS 7200 RPM
Network controller	1 × 10 Gb Ethernet
Notes	CPU con Intel's Hyper-Threading e QPI.

Hadoop cluster elementi di progettazione

- Selezione hardware
- **High Availability – I cluster non dovrebbero mai cadere**

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- **Sicurezza – I Cluster dovrebbero coprire tutti i livelli di sicurezza**
 - Autenticazione (autenticazione via HTTP, client ssl, proxy ...)
 - Autorizzazione (accesso ai file definito per utenti)
 - Accountability (records, logging, cronologia)
 - Criptaggio
 - Spostamento dati (RPC)

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- Sicurezza
- **Scalabilità – massimizzare prestazioni di rete e IO e minimizzare il livello fisico**

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- Sicurezza
- Scalabilità
- **Hadoop su macchine virtuali – elasticità, data isolation, massimizzazione dell'uso delle risorse hardware , improved multi-tenancy**

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- Sicurezza
- Scalabilità
- Hadoop su macchine virtuali
- **Dimesionamento del cluster**

Tasso di ingestione medio	1 TB	
Fattore di replicazione	3	copie per ogni blocco
Consumo giornaliero	3 TB	ingestione x replicazione
Disco per nuovo nodo	24 TB	$12 \times 2 \text{ TB SATA II HDD}$
Cartella temporanea per mapReduce	25%	Per i dati intermedi di mapReduce
Spazio utilizzabile sui nodi	18 TB	Bisogna riservare una parte per i dati temporanei
Crescita annua (lineare)	61 nodi / anno	ingestione x replicazione x 365 / spazio nodo
Crescita annua con incremento del 5%	81 nodi / anno	
Crescita annua con incremento del 10%	109 nodi / anno	

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- Sicurezza
- Scalabilità
- Hadoop su macchine virtuali
- Dimensionamento del cluster
- **Hostname e DNS**

Hadoop cluster elementi di progettazione

- Selezione hardware
- High Availability
- Sicurezza
- Scalabilità
- Hadoop su macchine virtuali
- Dimensionamento del cluster
- Hostname e DNS



Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

BIG DATA e Analisi dei Dati

Lezione 2.4 – Etica sul Big Data

Mauro Bellone,
Robotics and AI researcher

bellonemauro@gmail.com
www.maurobellone.com

Etica sui big data

Big data ethics aka simply data ethics refers to systemizing, defending, and recommending concepts of right and wrong conduct in relation to data, in particular personal data.

The screenshot shows the English Wikipedia article on "Big data ethics". The page has a dark theme. At the top, there's a navigation bar with "Article" and "Talk" buttons. Below the title "Big data ethics" is a green sidebar featuring a tree icon. The main content area starts with a note about the article being a personal reflection or essay. It then describes what big data ethics is, noting it's different from information ethics. A "Contents" box is on the right, listing sections like "Principles", "Personal Data of Children", and "Manifestos, declarations, and unions". The bottom section is titled "Principles" and lists three ownership principles. A yellow box highlights the "Languages" section at the bottom left of the page.

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate
Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file
Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item
Print/export
Download as PDF
Printable version
Languages
فارسی
Français
Português
Edit links

Article Talk

Big data ethics

This article is written like a personal reflection, personal essay, or argumentative essay that states a Wikipedia editor's personal feelings or presents an original argument about a topic. Please help improve it by rewriting it in an encyclopedic style. (December 2019) (Learn how and when to remove this template message)

Big data ethics also known as simply data ethics refers to systemizing, defending, and recommending concepts of right and wrong conduct in relation to data, in particular personal data. It describes this large amount of data that is so voluminous and complex that traditional data processing methods are inadequate. Internet-connected health devices have triggered a data deluge that will reach the exabyte range by 2020. Big data ethics is different from information ethics because the focus of information ethics is more on the use of data by individuals, governments, and large corporations.

Contents [hide]

1 Principles

- 1.1 Ownership
- 1.2 Transaction transparency
- 1.3 Consent
- 1.4 Privacy
- 1.5 Currency
 - 1.5.1 How Much is Data Worth?
- 1.6 Openness

2 Personal Data of Children

3 The role of institutions

- 3.1 Nation states
- 3.2 Banks

4 Relevant news items about data ethics

5 Relevant legislation about data ethics

6 Manifestos, declarations, and unions

7 See also

8 Footnotes

9 References

Principles

Data ethics is concerned with the following principles. [original research?]

1. Ownership - Individuals own their own data.
2. Transaction transparency - If an individual's personal data is used, they should have transparency.
3. Consent - If an individual or legal entity would like to use personal data, one needs informed consent.

Voce NON disponibile in lingua italiana !!!

Etica sui big data: Principi

- **Proprietà:** A chi appartengono i dati?

¹ “*Individuals own their own data.*“

Etica sui big data: Principi

- **Trasparenza della transazione:** Nel momento in cui c'è un effettivo utilizzo dei dati personali, l'utente dovrebbe avere accesso agli algoritmi (progetto) utilizzati per il trattamento dei dati

*“If an individual personal data is used, they should have transparent access to the **algorithm design** used to generate aggregate data sets”*

Etica sui big data: Principi

- **Consenso:** l'utilizzatore dei dati deve avere il consenso del proprietario dei dati

*“If an individual or legal entity would like to use personal data, one needs **informed and explicitly expressed consent** of what personal data moves to **whom, when, and for what purpose** from the owner of the data.”*

Etica sui big data: Principi

- **Privacy:** I dati non dovrebbero essere comunicati a terzi senza esplicito consenso del proprietario

“If data transactions occur all reasonable effort needs to be made to preserve privacy.”

Etica sui big data: Principi

- **Privacy:** I dati non dovrebbero essere comunicati a terzi senza esplicito consenso del proprietario

“If data transactions occur all reasonable effort needs to be made to preserve privacy.”

“No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.”

United Nations Declaration of Human Rights Article 12

Etica sui big data: Principi

- **Moneta:** La comunicazione dei dati è la moneta di scambio per l'utilizzo di servizi gratuiti

*“Individuals should be aware of **financial transactions** resulting from the use of their personal data and the scale of these transactions”*

Etica sui big data: Principi

- **Apertura:** I dati dovrebbero essere aperti



openknowledge



OpenAI

