Hadoop Distributed File System
HDFS

Sistemi Distribuiti e Cloud Computing

Obiettivi HDFS

- File system distribuito molto grande
  - 10.000 nodi, 100 milioni di file, 10 PB
- I file sono replicati per gestire guasti hardware (fault tolerant)
  - Rileva i guasti e ripristina lo stato del sistema
- Ottimizzato per l'elaborazione in batch di Big Data.
- Locazioni dei dati esposte in modo che i processi possano spostarsi dove risiedono i dati.
- Fornisce una larghezza di banda aggregata molto elevata.

Obiettivi HDFS

- HDFS usa: File, directories e comandi shell.

- Modello ***Write-once-read-many*** (con possibilità di append).
- I file in HDFS sono write-once e quindi hanno uno 'scrittore' alla volta, ma molti lettori concorrenti.

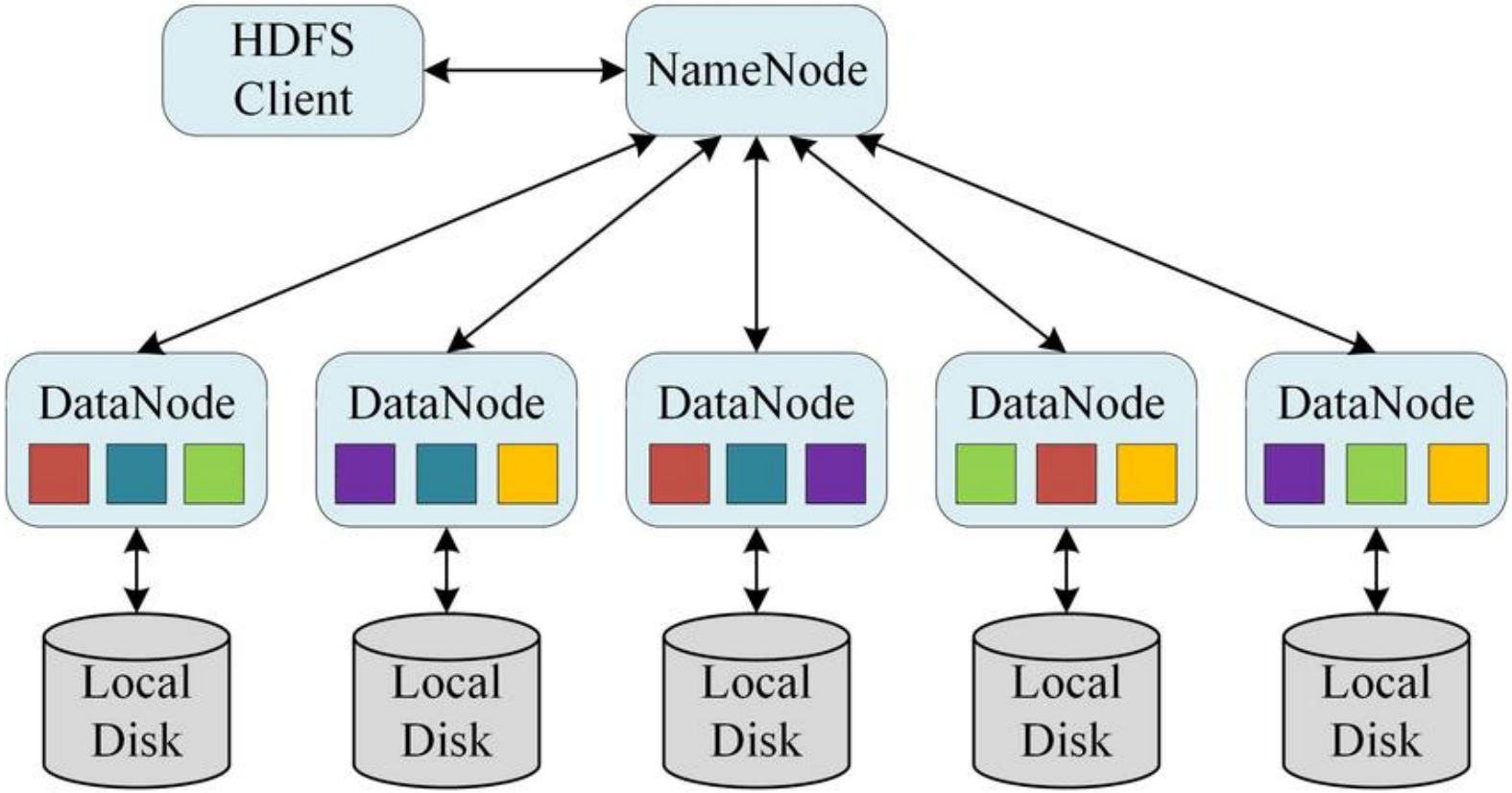
- Questa ipotesi semplifica i problemi di coerenza dei dati e consente l'accesso ai dati ad alta velocità.

- Un'applicazione MapReduce o un web crawler si adatta perfettamente a questo modello.

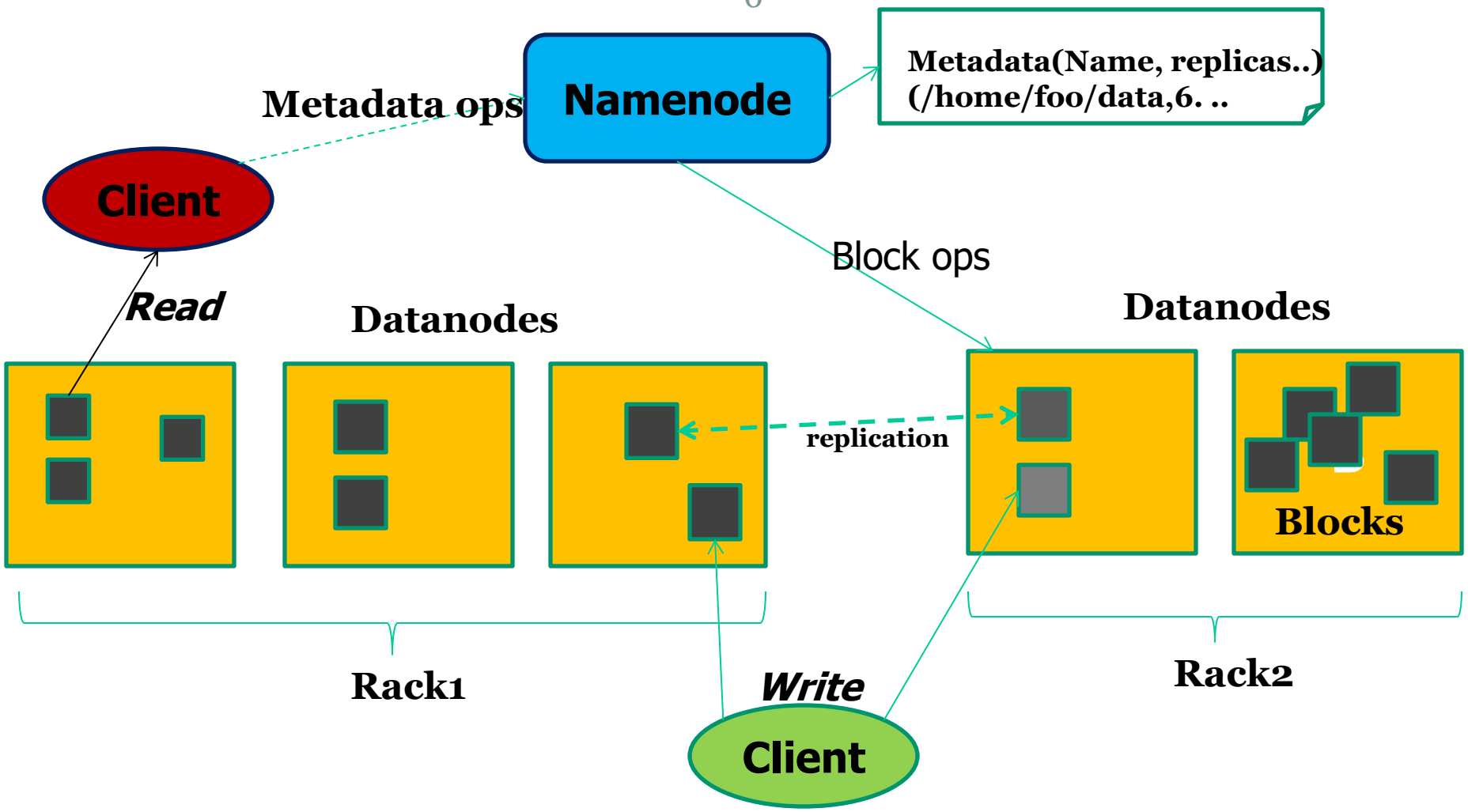
Architettura HDFS

- Architettura master/slave:
  - Il cluster HDFS è costituito da un singolo **Namenode**, un server master che gestisce lo spazio dei nomi del file system e regola l'accesso ai file da parte dei client.
  - Ci sono un certo numero di **DataNode**, di solito uno per nodo in un cluster.
  - I **DataNode** gestiscono l'archiviazione nei nodi su cui vengono eseguiti.
  - I **DataNode** servono le richieste di lettura, scrittura, eseguono la creazione di blocchi, l'eliminazione e la replica su richiesta del **Namenode**.
  - HDFS espone uno spazio dei nomi del file system e ogni file viene suddiviso in uno o più blocchi e una serie di blocchi viene archiviata nei **DataNodes**.

Architettura HDFS



Architettura HDFS



Blocchi HDFS

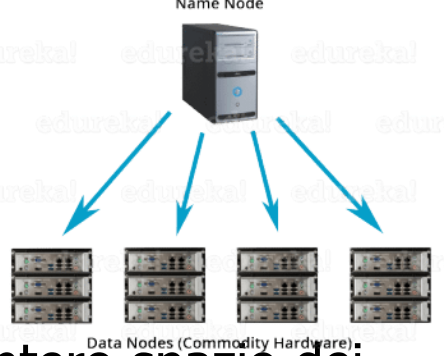
- HDFS è progettato per archiviare file molto grandi su macchine in cluster di grandi dimensioni (data center/cloud/HPC).
- Ogni file è una sequenza di blocchi. Tutti i blocchi nel file tranne l'ultimo hanno la stessa dimensione.
- I blocchi sono replicati per la tolleranza ai guasti. La dimensione del blocco e le repliche sono configurabili per file.
- Il Namenode riceve un *Heartbeat* e un *BlockReport* da ogni DataNode nel cluster.
  - L'Heartbeat implica che il Datanode è attivo.
  - Il BlockReport contiene la lista di tutti i blocchi sul Datanode.

File e Replica selection

- Una dimensione tipica del blocco è 64 MB (o anche 128 MB).
- Un file viene quindi diviso in blocchi da 64/128 MB e archiviato.

- Selezione delle repliche per l'operazione di LETTURA:
  - HDFS tenta di ridurre al minimo il consumo di larghezza di banda e la latenza.
  - Se è presente una replica sul nodo Client, è preferibile.
  - Il cluster HDFS può estendersi su più data center: la replica nel data center locale è preferita a quella remota.
- Le repliche sono poste su singoli rack di un cluster.

Namenode



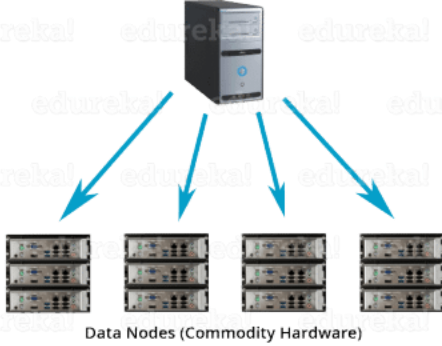
- Il ***Namenode*** mantiene in memoria l'immagine dell'intero spazio dei nomi del file system e del file *Blockmap* (che contiene le info sui blocchi di un ***Datanode***).

- 4 GB di RAM locale sono sufficienti per supportare le strutture dati che rappresentano l'enorme numero di file e directory.

- Quando il ***Namenode*** si avvia, ottiene *FsImage* (metadati di directory e files) e *Editlog* (metadati temporanei) dal suo file system locale, aggiorna *FsImage* con le informazioni di *EditLog* e quindi memorizza una copia di *FsImage* sul file system come checkpoint.

- Effettua il checkpoint periodico. In modo che il sistema possa ripristinare l'ultimo stato di checkpoint in caso di arresto anomalo.

Datanode



- Un ***Datanode*** memorizza i dati nei file nel proprio file system locale.
- Un ***Datanode*** non ha alcuna conoscenza del filesystem HDFS.
- HDFS Memorizza ogni blocco di dati HDFS in un file separato.
- Un ***Datanode*** non crea tutti i file nella stessa directory. Utilizza l'euristica per determinare il numero ottimale di file per directory e crea le directory in modo opportuno.
- Quando il filesystem si avvia, genera un elenco di tutti i blocchi HDFS e invia questo rapporto a ***Namenode*** tramite un *Blockreport*.

Protocolli di Comunicazione

- Tutti i protocolli di comunicazione HDFS sono costruiti sul protocollo TCP/IP.

- Un client stabilisce una connessione a una porta TCP configurabile sulla macchina ***Namenode*** e comunica con il *ClientProtocol*.
- I ***Datanode*** comunicano con il ***Namenode*** usando il protocollo del *Datanode (DN protocol)*.

- L'astrazione RPC incapsula sia il protocollo *ClientProtocol* che il protocollo *Datanode*.

- Il ***Namenode*** è semplicemente un server e non avvia mai una richiesta; risponde solo alle richieste RPC emesse da ***DataNodes*** o client.

Datanode Failure

- Una partizione di rete può causare la perdita di connettività di un sottoinsieme di ***Datanode*** con il ***Namenode***.
- Il ***Namenode*** rileva questa condizione dall'assenza di un messaggio *Heartbeat*.
- Il ***Namenode*** ricorda i ***Datanode*** senza *Hearbeat* e non invia loro alcuna richiesta di I/O.
- Tutti i dati registrati nel ***Datanode*** guasto non sono disponibili per HDFS.

Gestione Repliche

- Il crash di un ***Datanode*** può far sì che il fattore di replica di alcuni blocchi scenda al di sotto del valore specificato.

- La necessità di una nuova replicazione può sorgere perchè:
  - Un ***Datanode*** potrebbe non essere disponibile,
  - Una replica potrebbe danneggiarsi,
  - Un disco rigido su un ***Datanode*** potrebbe non funzionare,
  - Il fattore di replica sul blocco può essere stato aumentato.

Data Integrity

- Consideriamo una situazione in cui un blocco di dati recuperato da un ***Datanode*** arrivi danneggiato.
- Questo danno può verificarsi a causa di errori in un dispositivo di storage, errori di rete o problemi software.
- Un client HDFS crea il *checksum* di ogni blocco del suo file e lo archivia in file nascosti nello spazio dei nomi HDFS.
- Quando un client recupera il contenuto del file, verifica che i *checksum* corrispondenti siano corretti. Se non c'è corrispondenza, il client può recuperare il blocco da una sua replica.

Interfacce HDFS

- HDFS fornisce API Java da utilizzare per le applicazioni.
- L'accesso Python è utilizzato anche in molte applicazioni.
- È inoltre disponibile un wrapper del linguaggio C per l'API Java.
- È possibile utilizzare un browser HTTP per accedere ai file di un'istanza HDFS.