

HDFS

Hadoop Distributed File System



- Popescu Vlad-Gabriel
- Lupu Mugurel-Orlando



Informatii Generale

Apache Hadoop este un framework open-source folosit pentru sisteme distribuite de stocare si pentru operatiuni cu date de dimensiuni mari, folosind paradigma Map Reduce. Pentru a realiza acest lucru sunt folosite clustere formate din mai multe calculatoare ce isi impart resursele

Hadoop a fost creat folosind Google MapReduce si Google File System, fiind unul dintre flag-shipurile celor de la Apache. Deasemenea, Yahoo a fost unul dintre principalii contribuitori, implementarea fiind in Java.



Arhitectura

Hadoop reprezinta un pachet de utilitare numit "Hadoop Common". Acesta inglobeaza toate aplicatiile hadoop si contine fisierele in format JAR, documentatie si fisierele cu contribuitorii

Fiecare sistem de fisiere ce ruleaza Hadoop va furniza acestuia date despre retea, in vederea optimizarii activitatii executate pe acel nod, iar in caz de esec sa distribuie procesarea in cadrul aceluia switch/rack/router pentru a reduce traficul pe magistrala principala. Deasemenea se utilizeaza si vederea replicarii datelor pe mai multe retele distincte, pentru a minimiza pierderile de informatii in cazul unei defectiuni fizice.



Hardware Failure

Hardware failure is the norm rather than the exception. An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data. The fact that there are a huge number of components and that each component has a non-trivial probability of failure means that some component of HDFS is always non-functional. Therefore, detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

Streaming Data Access

Applications that run on HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. POSIX imposes many hard requirements that are not needed for applications that are targeted for HDFS. POSIX semantics in a few key areas has been traded to increase data throughput rates.

Large Data Sets

Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance.



Simple Coherency Model

HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed. This assumption simplifies data coherency issues and enables high throughput data access. A MapReduce application or a web crawler application fits perfectly with this model. There is a plan to support appending-writes to files in the future.

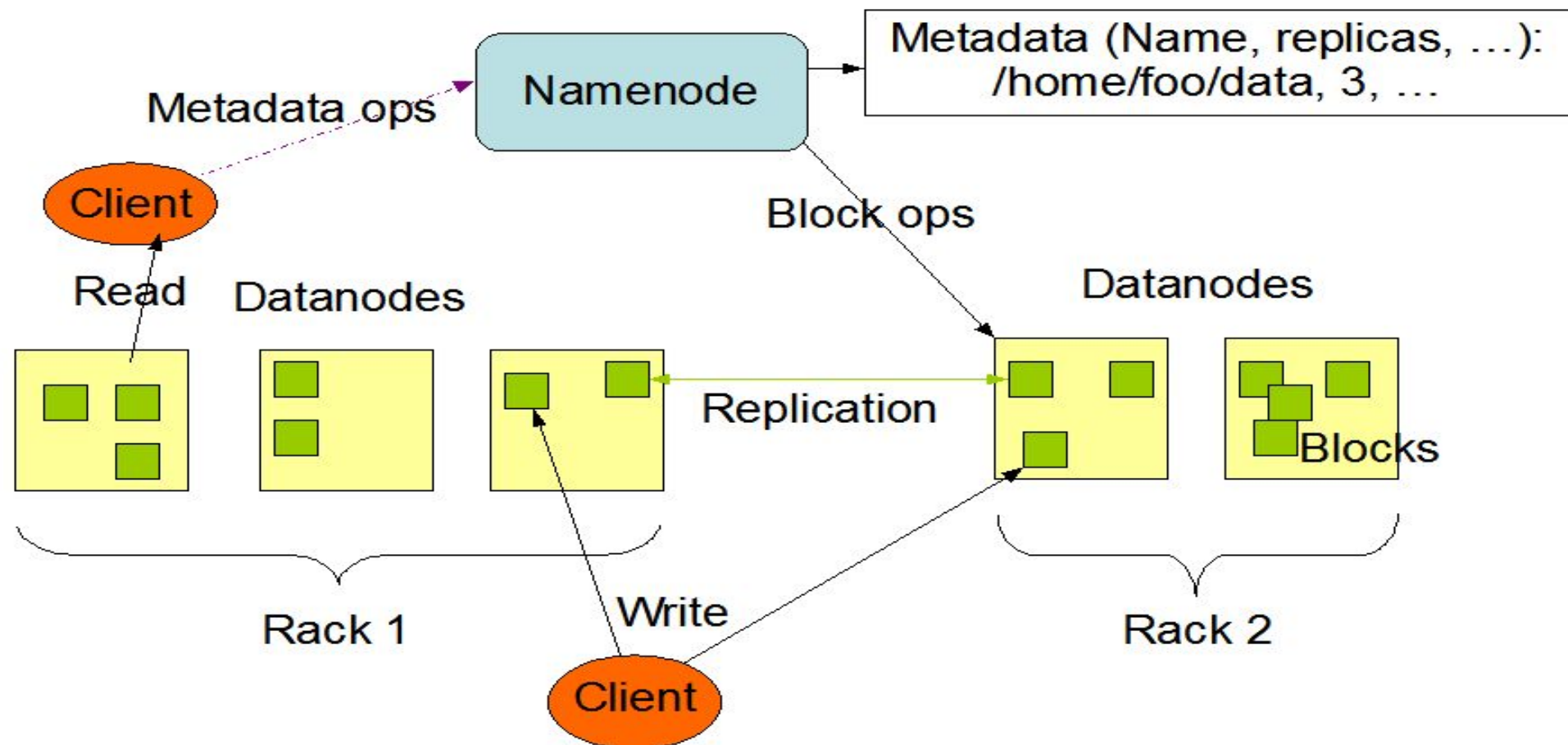
“Moving Computation is Cheaper than Moving Data”


A computation requested by an application is much more efficient if it is executed near the data it operates on. This is especially true when the size of the data set is huge. This minimizes network congestion and increases the overall throughput of the system. The assumption is that it is often better to migrate the computation closer to where the data is located rather than moving the data to where the application is running. HDFS provides interfaces for applications to move themselves closer to where the data is located.

Portability Across Heterogeneous Hardware and Software Platforms

HDFS has been designed to be easily portable from one platform to another. This facilitates widespread adoption of HDFS as a platform of choice for a large set of applications.

HDFS Architecture





HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

