Distributed Data Management with Google File System

Prof. Pramod Bhatotia
https://dse.in.tum.de/bhatotia/

From the last class: How big is "Big Data"?

processes 20 PB a day (2008)
crawls 20B web pages a day (2012)

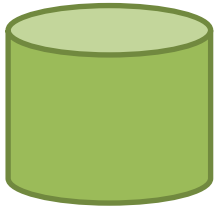
>10 PB data, 75B DB calls per day (6/2012)

>100 PB of user data + 500 TB/day (8/2012)

S3: 449B objects, peak 290k request/second (7/2011)
1T objects (6/2012)

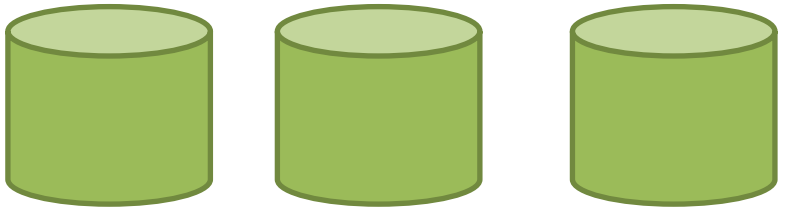
Distributed systems for "Big Data" management

Single disk



Bandwidth: ~180 MB/sec,
Read time 1PB: ~65 days!

Multiple disks



Aggregate bandwidth:
~ No. of disks*180 MB/sec

In today's class

How to manage "Big data" in distributed setting?

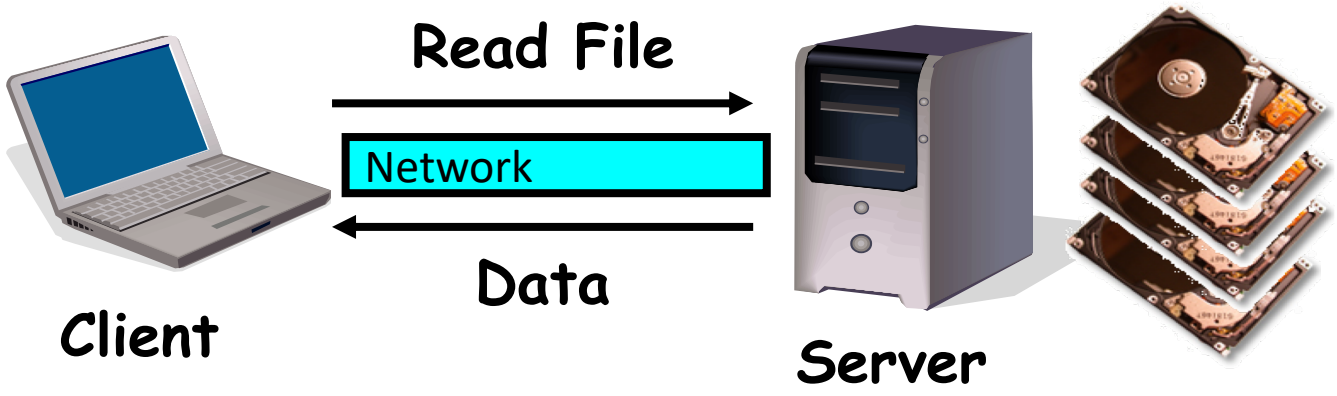
- Google File System (GFS)
  - Open source: Hadoop Distributed File System (HDFS)

Distributed file system

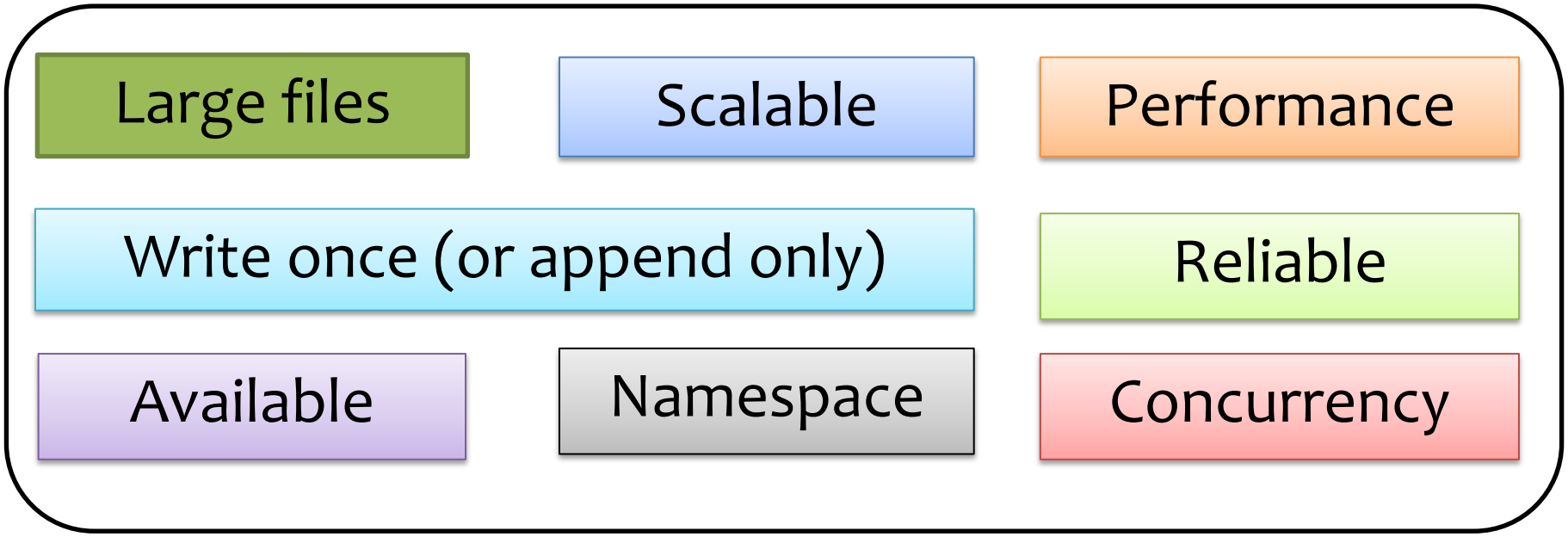
- From Google search:

  "A distributed file system is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer."

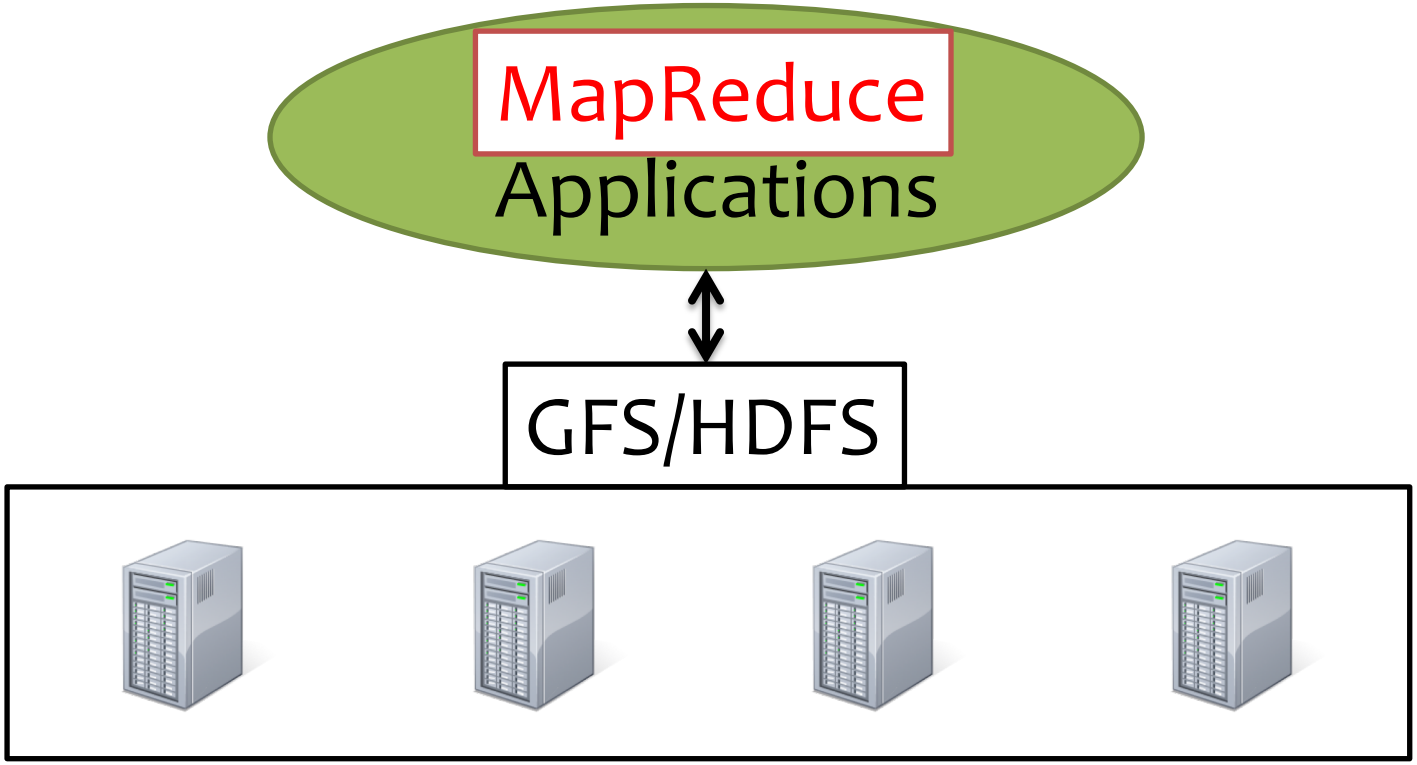
- E.g.: NFS, AFS



Key design requirements



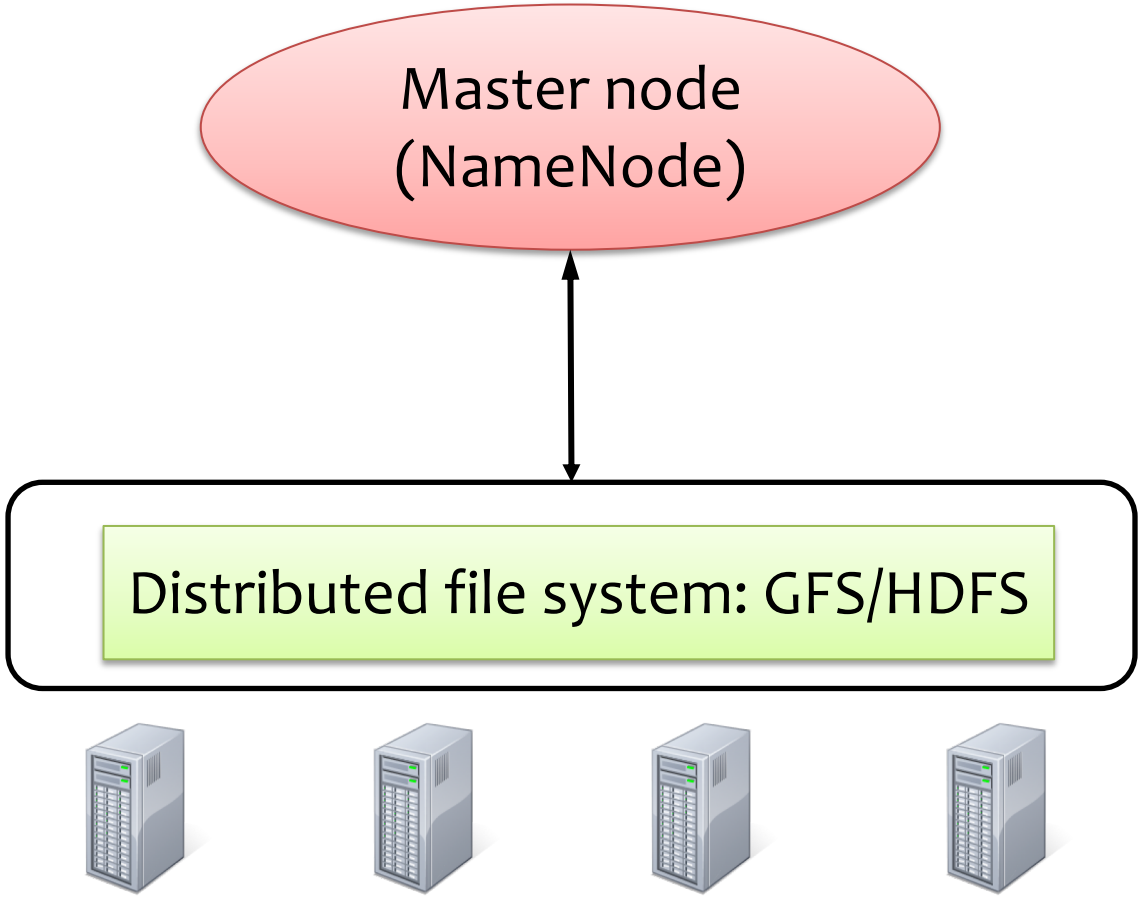
Beauty of GFS/HDFS



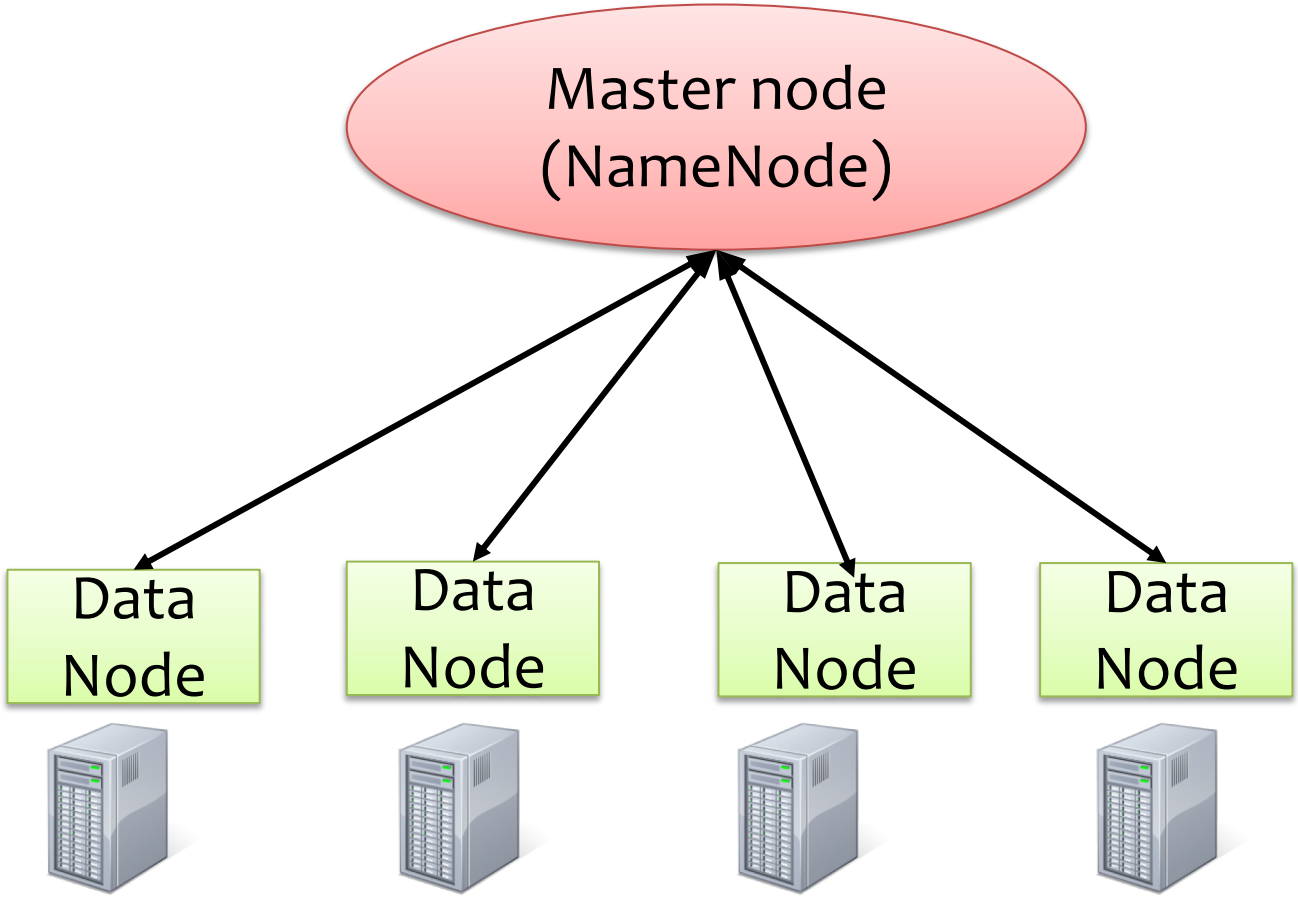
Interface/API

- Simple shell interface
  - $ hadoop dfs -<fs commands> <arguments>
  - <fs commands>: mv, cp, chmod, copyFromLocal, copyToLocal, rm, du, etc.
  - Additional commands for snapshots, appends
- Similar programming APIs to access HDFS
  - Not compatible w/ POSIX API; HDFS only allows appends

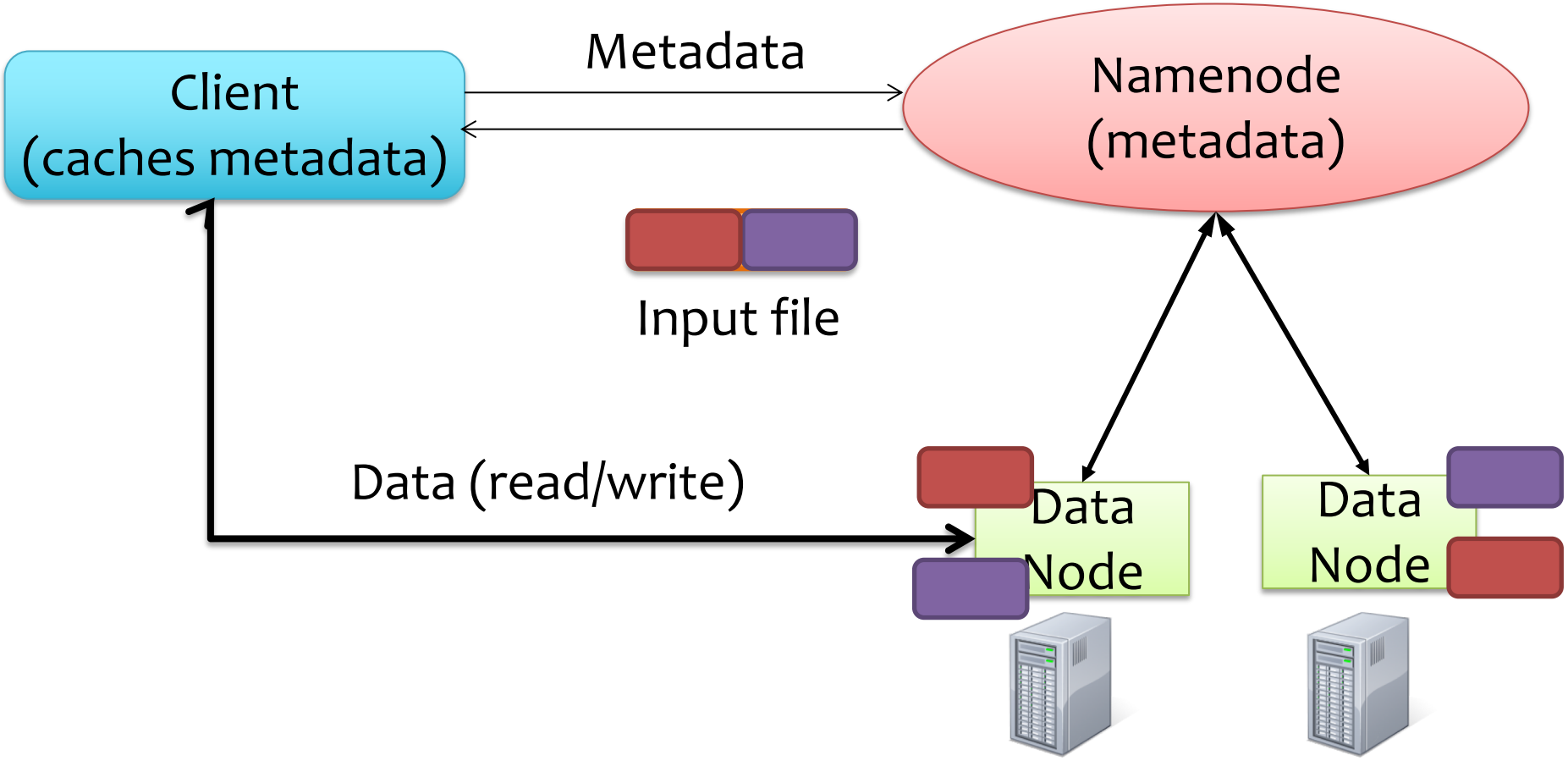
Architecture



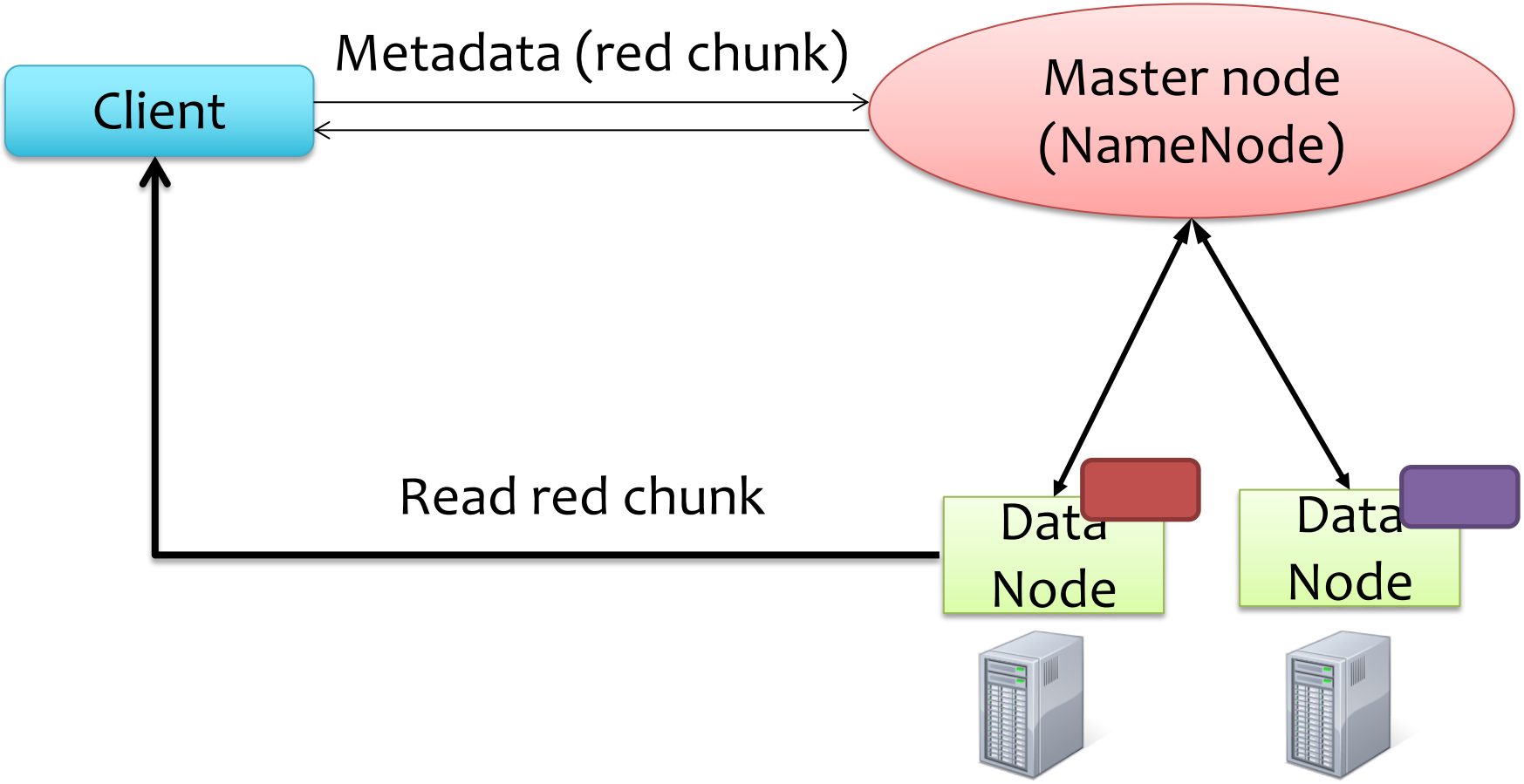
Architecture



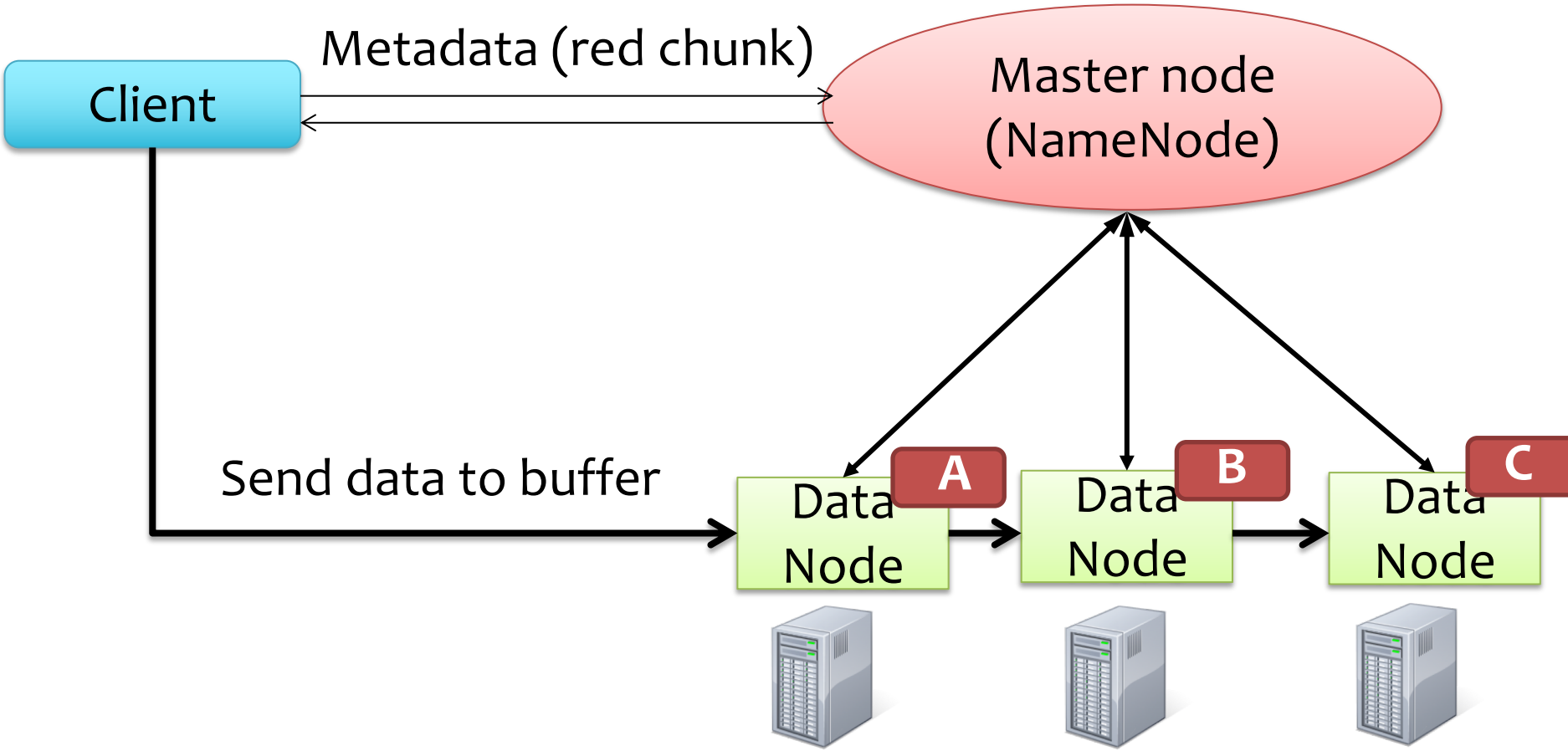
Basic functioning



Read operation



Write operation

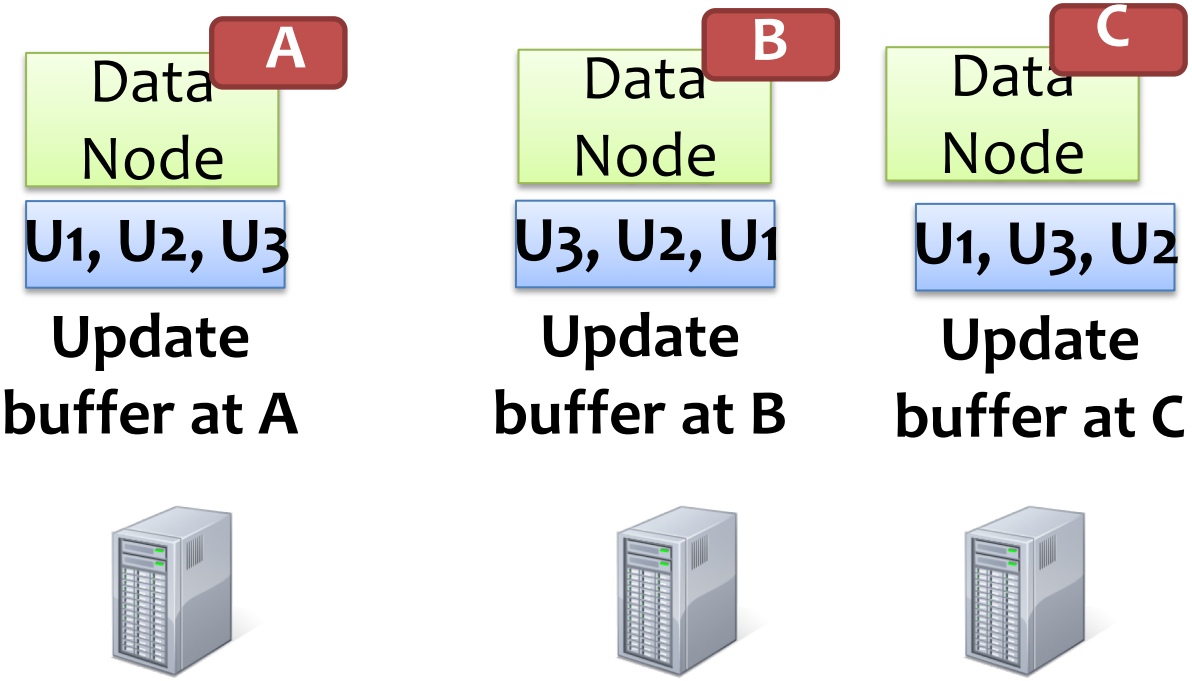


Commit order

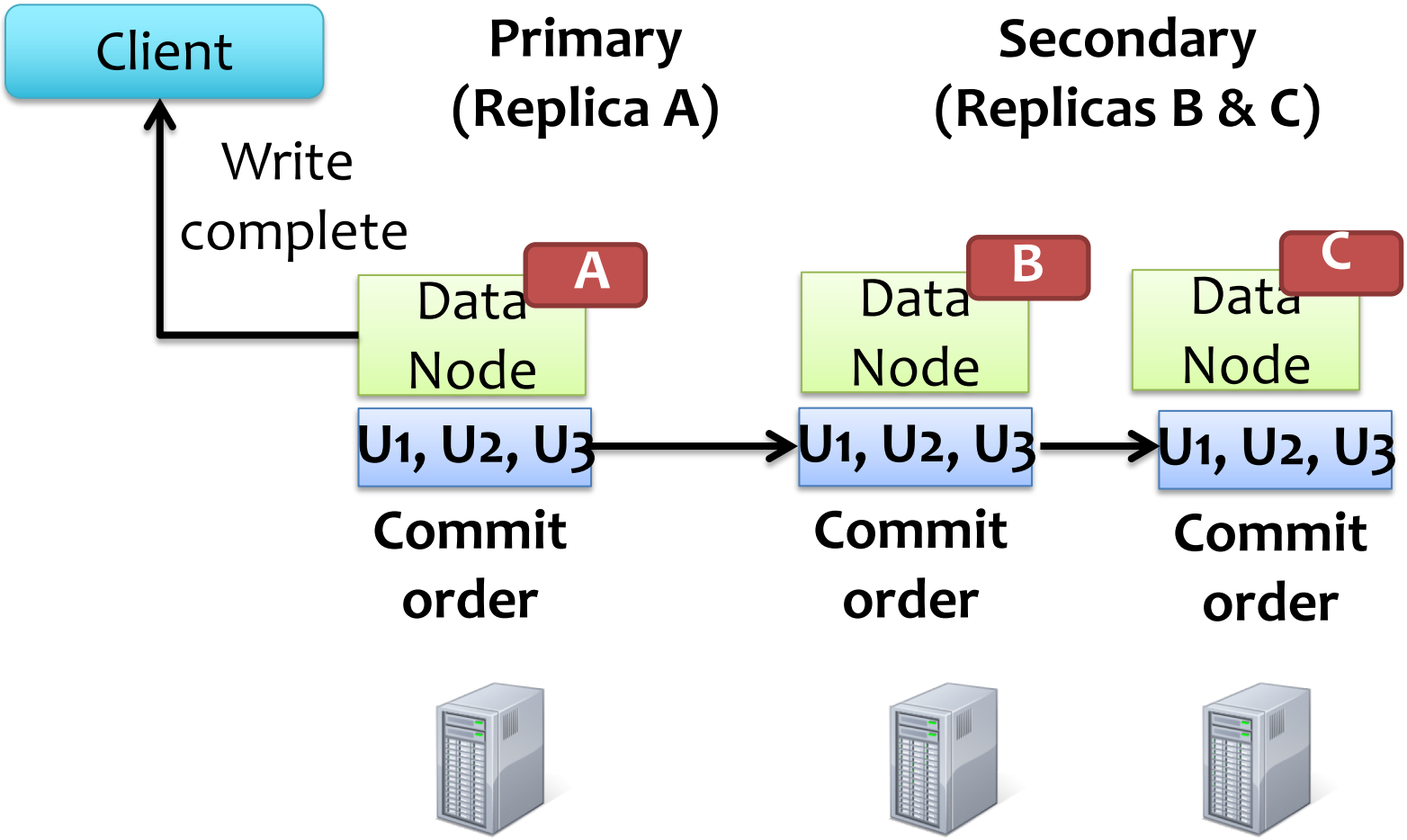
Using lease management by the master

Primary (Replica A)

Secondary (Replicas B & C)



Primary commit order



Consistency semantics

- Updates (only for GFS)
  - File region may end up containing mingled fragments from different clients
  - E.g., writes to different chunks may be ordered differently by their different primary replica
  - Thus, writes are consistent but undefined in GFS

- Appends
  - Append causes data to be appended atomically at least once
  - Offset chosen by HDFS, not by the client

Discussion: Other design details

- Chunk replication: placement and re-balancing
- Namespace management & locking
- Garbage collection (lazy)
- Data integrity
- Snapshots (using copy-on-write)
- Master replication

References

- Compulsory reading: GFS [SOSP'03]
  - Google File System paper https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf

- Recommended reading: Tachyon [SoCC'13]
  - Distributed in-memory file system for Spark
- Resources: www.hadoop.apache.org