Beyond Relational

Database with

Large datasets / **phat** throughoutput

Share nothing ...

- ... but the database
- Ruby on Rails, PHP : no shared memory

Optimizing For High Traffic

- Replicating
- Dropping constraints
- Denormalizing
- Horizontal partitioning : “Sharding”
- Caching
- And afterwards ?

From the trenches

- SecondLife
- Flickr
- Craiglist

Common Behavior

- All started from a single database (sometimes on a single host).
- Design evolved to fit increasing traffic.

Second Life (mysql)

- No clustering
- Sharding
- With a central database holding metadata about location
- HTTP based communication between components

Flickr (mysql)

- Started a vertical cluster
- Hit write performance wall
  - cluster writes as slowly as the slowest machine
- Divided data into shards
- Unique data stored : 935 GB
- Total duplicated : 3TB

Flickr (2)

- Data is federated
  - organized around the user
  - comments from one user to another are duplicated for both users
- Heavy caching

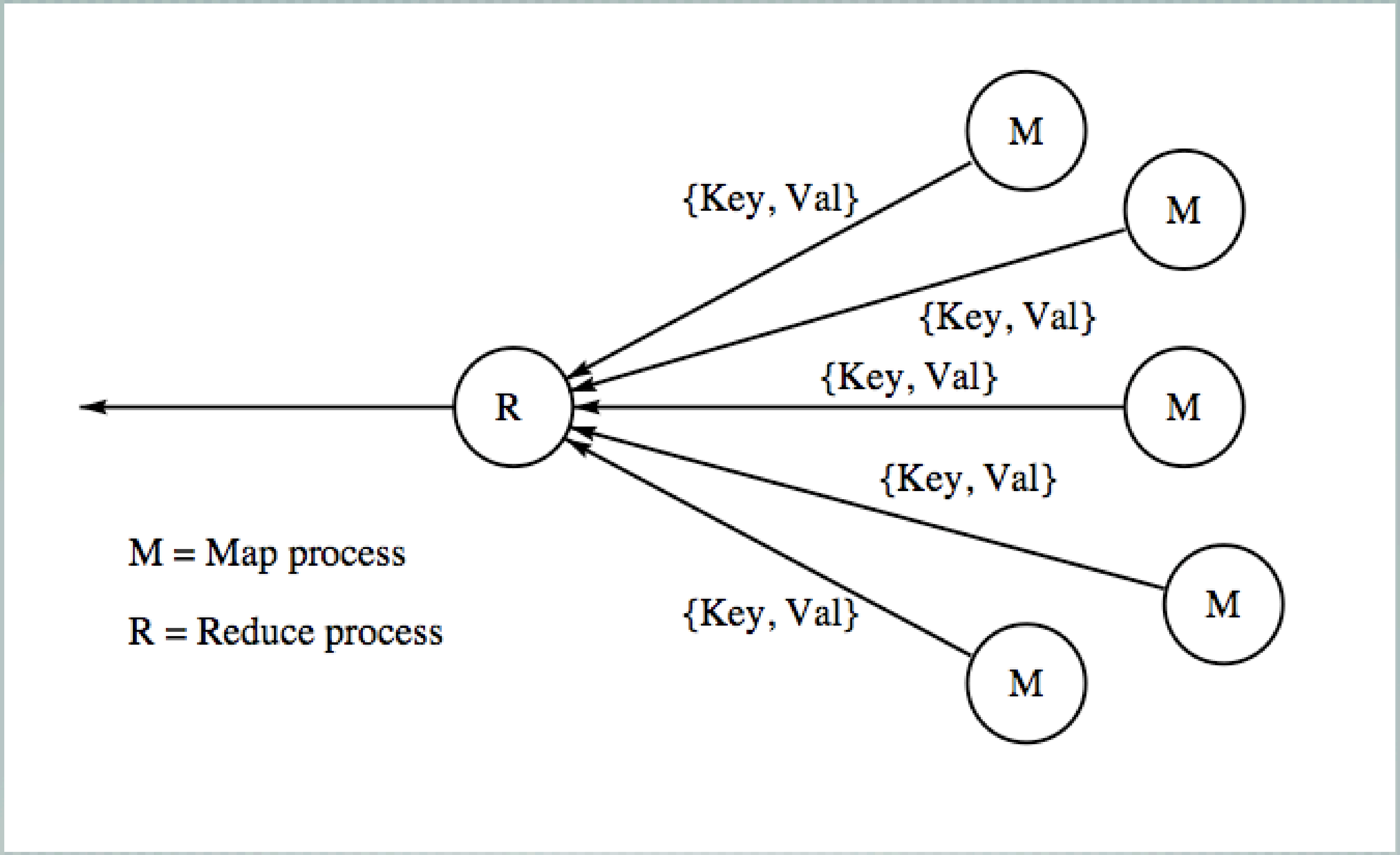
Craigslist

- MySQL on 64-bit Linux w/ 14 drives and 16GB RAM
- Classified DB : 12 slaves and 1 master.
  - 114 GB with a 56 million line table.
- SearchDB : for indexing, 16 servers in 4 clusters
  - host/table found in software

Google

- MapReduce
- GFS
- BigTable
- Sawzall

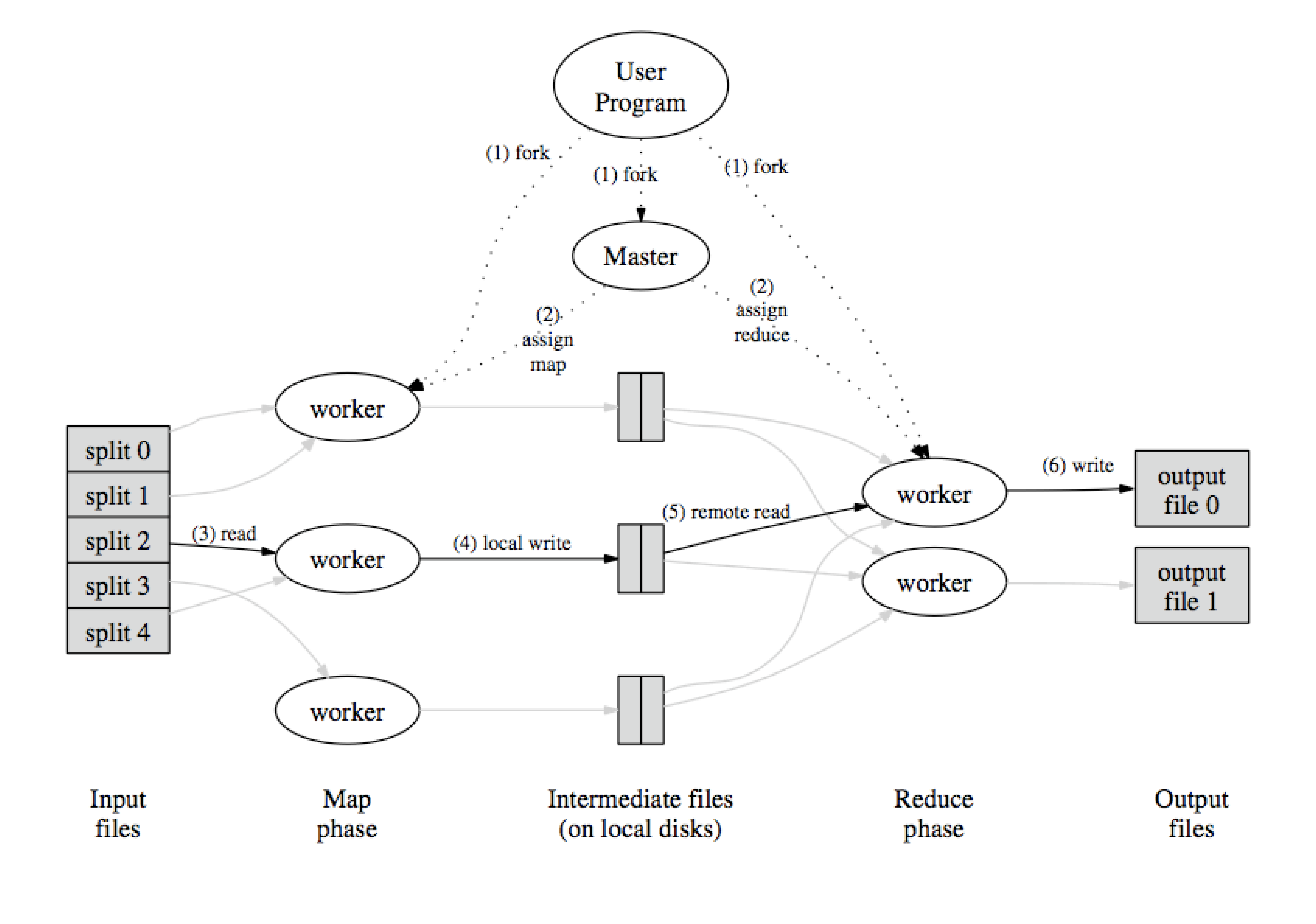
MapReduce


M
M
{Key, Val}
{Key, Val}
{Key, Val}
R
M
{Key, Val}
M = Map process
R = Reduce process
{Key, Val}
M
M

MapReduce : example

- 20 billion documents (avg size : 20 KB)
- Count each word occurence
- 400TB of data : 4 month's work on 1 machine

MapReduce : principles



MapReduce

- Used for indexing for Google WebSearch
- But also for Google Zeitgeist, Google News, Google Earth
- Since Feb. 2003
- by Jeffrey Dean and Sanjay Ghemawat

MapReduce Features

- Load Balancing
- Fault tolerance
- Locality
- Backup tasks

Load Balancing

- More tasks than machines
- New task assigned at end of previous task
- Faster machines do more work

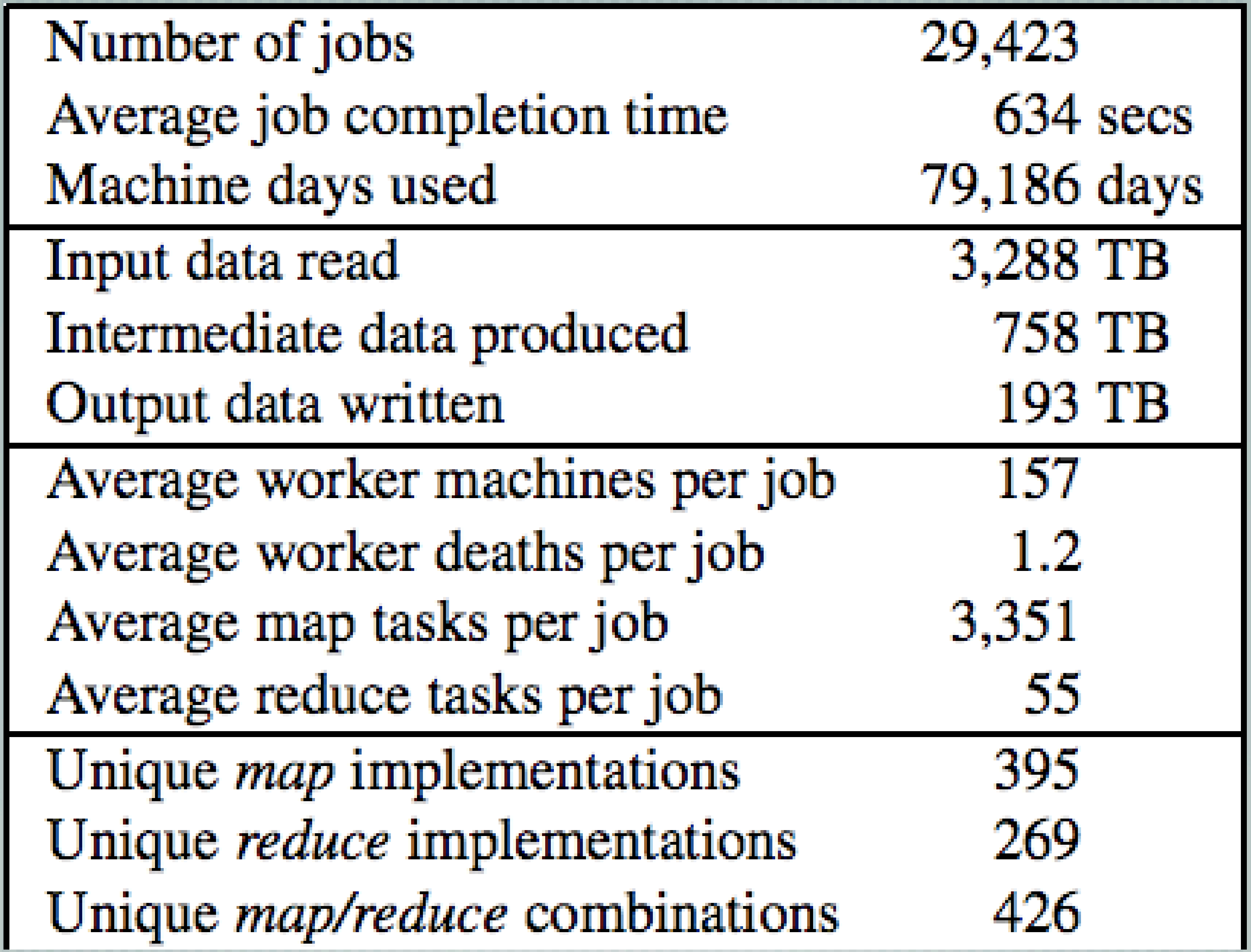
Fault tolerance

- Machine failure must be handled
- Master pings workers
- Worker set dead if does not respond several times
- Master logs its scheduling state, in case of master crash

Locality

- Input data (managed by GFS) is stored on the same machine executing the work
- For a Map task the Master will try to locate the servers where data is stored
- Master will schedule the Map task on this server

Usage for Aug 2004



Number of jobs	29,423
Average job completion time	634 secs
Machine days used	79,186 days
Input data read	3,288 TB
Intermediate data produced	758 TB
Output data written	193 TB
Average worker machines per job	157
Average worker deaths per job	1.2
Average map tasks per job	3,351
Average reduce tasks per job	55
Unique *map* implementations	395
Unique *reduce* implementations	269
Unique *map/reduce* combinations	426

GFS

- Google File System
- Distributed file system
- Each chunk (64MB) is stored in 3 locations
- Masters are replicated
  - State is distributed through log files

Architecture

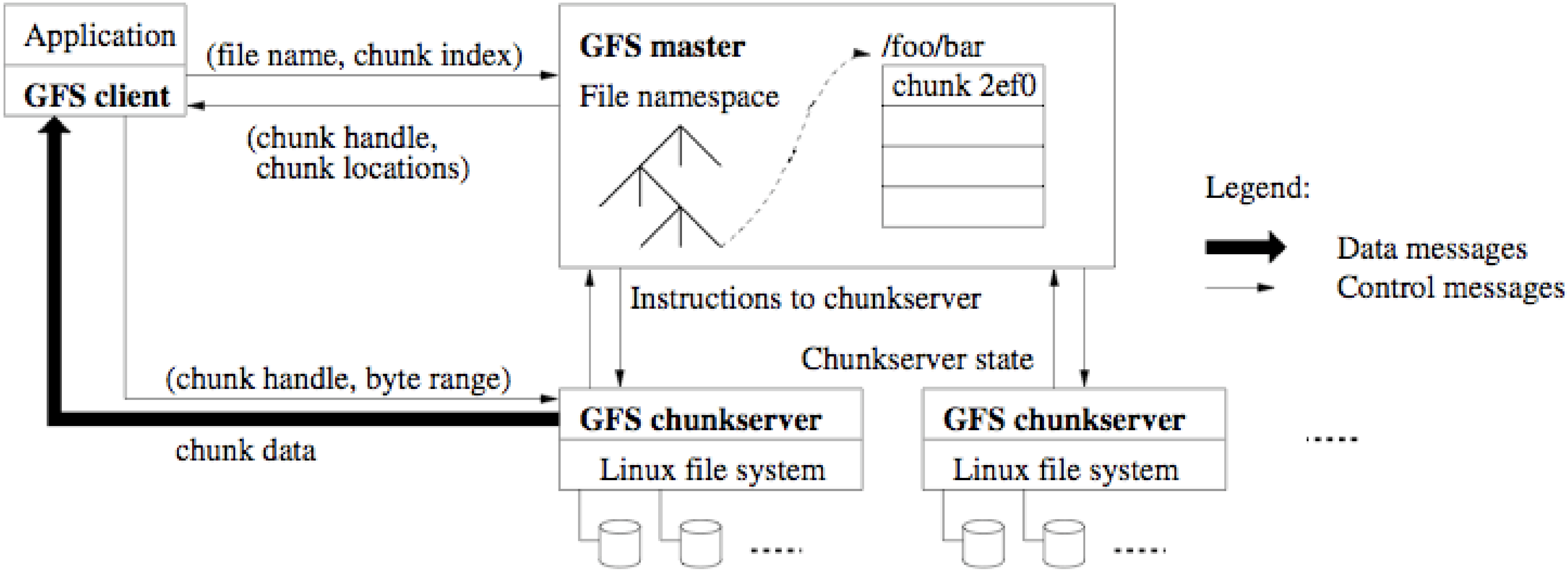
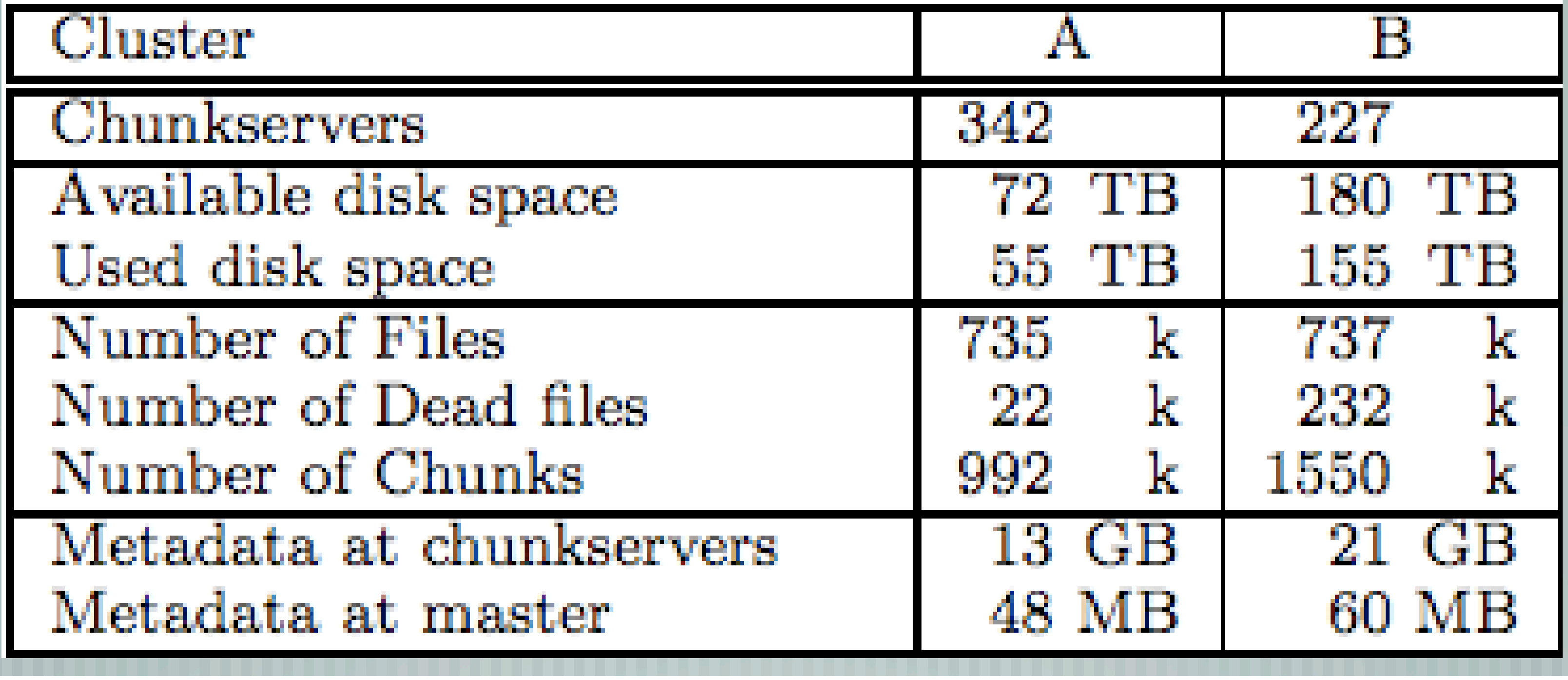
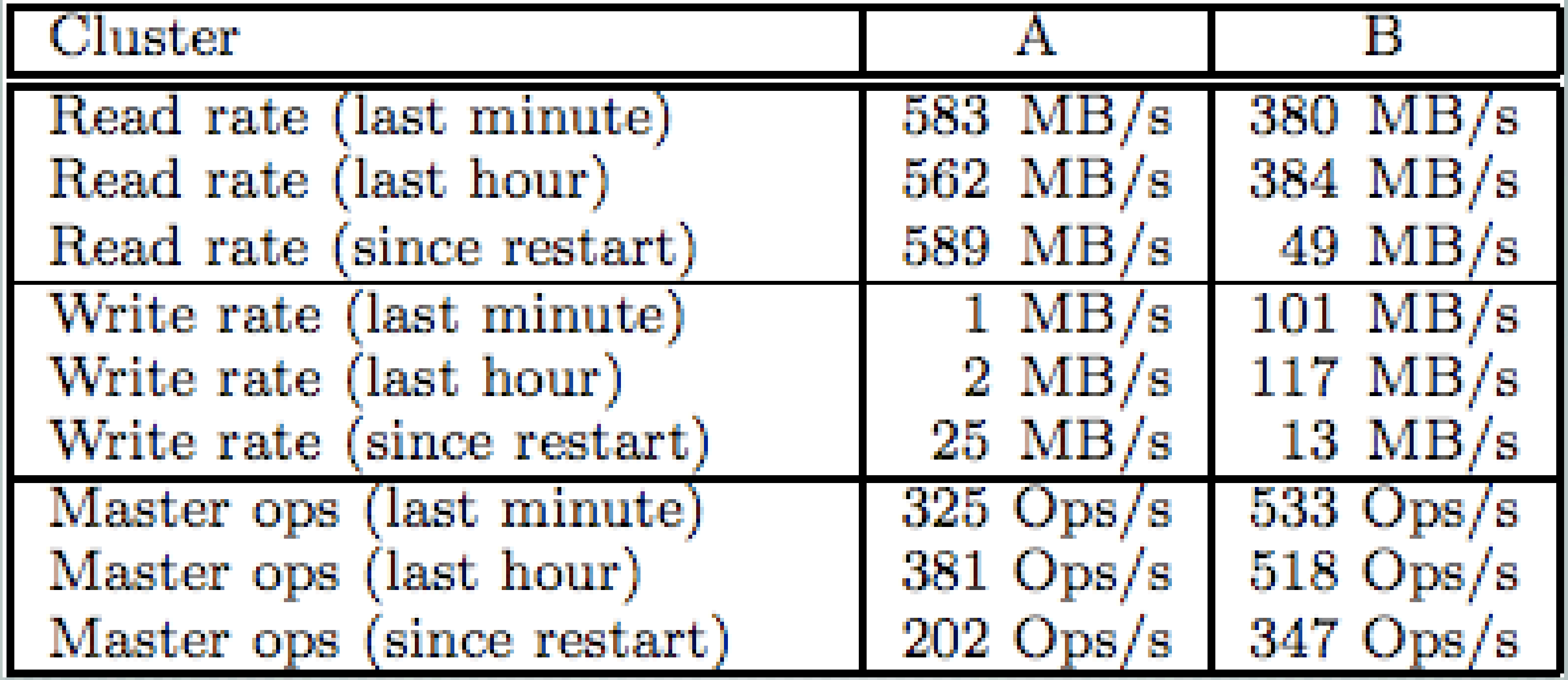


Figure 1: GFS Architecture

Performance



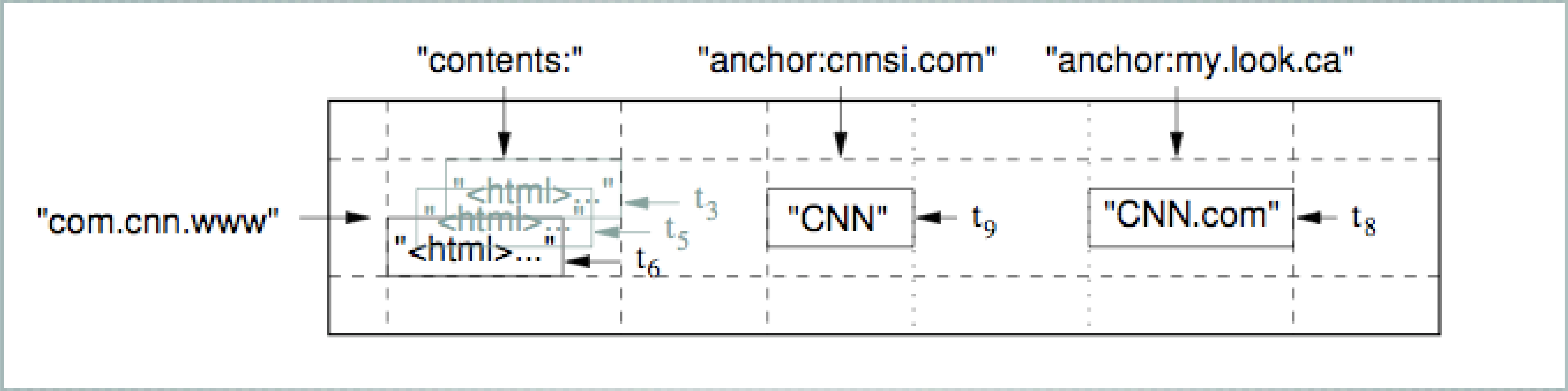
Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB



Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

BigTable

- Sparse Distributed Persistent Multi-dimensional Sorted Map
- (row:string, column:string, time:int64) -> string



BigTable

- Distributed locking and configuration done by Chubby
- A BigTable is distributed on tablet servers running GFS

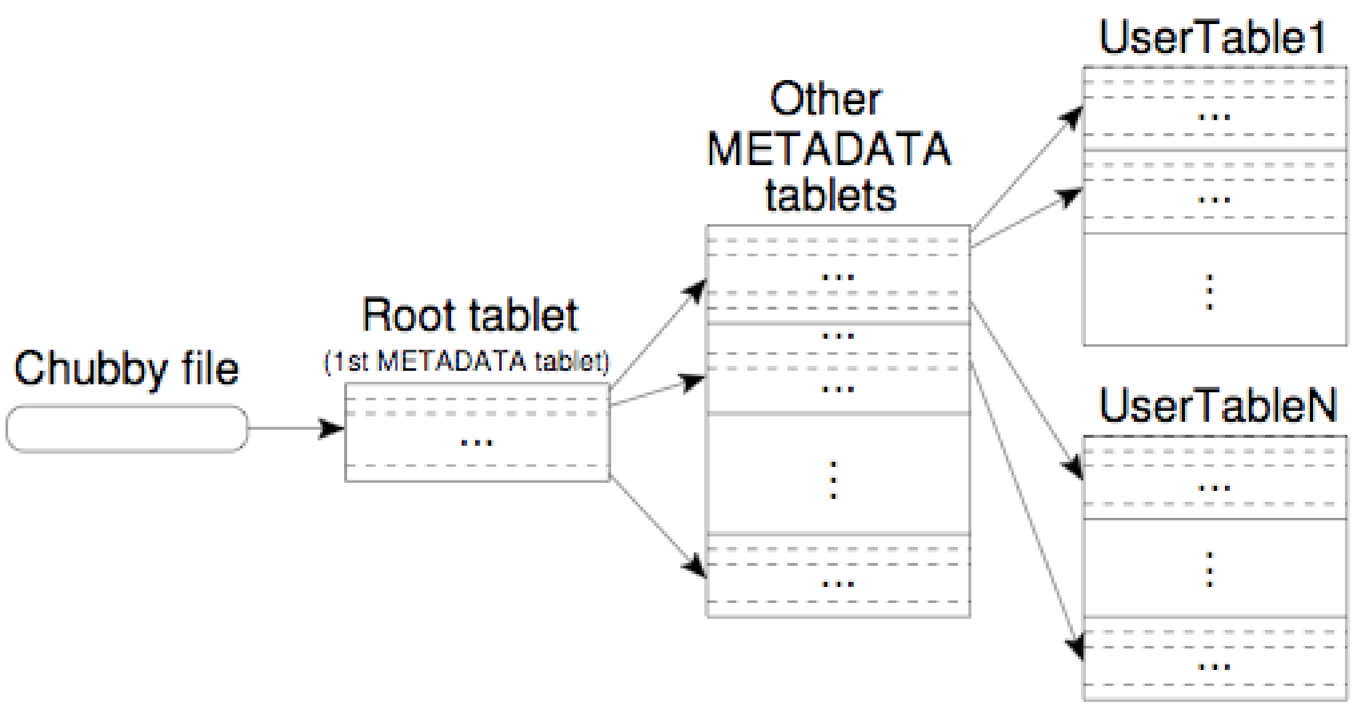


Figure 4: Tablet location hierarchy.

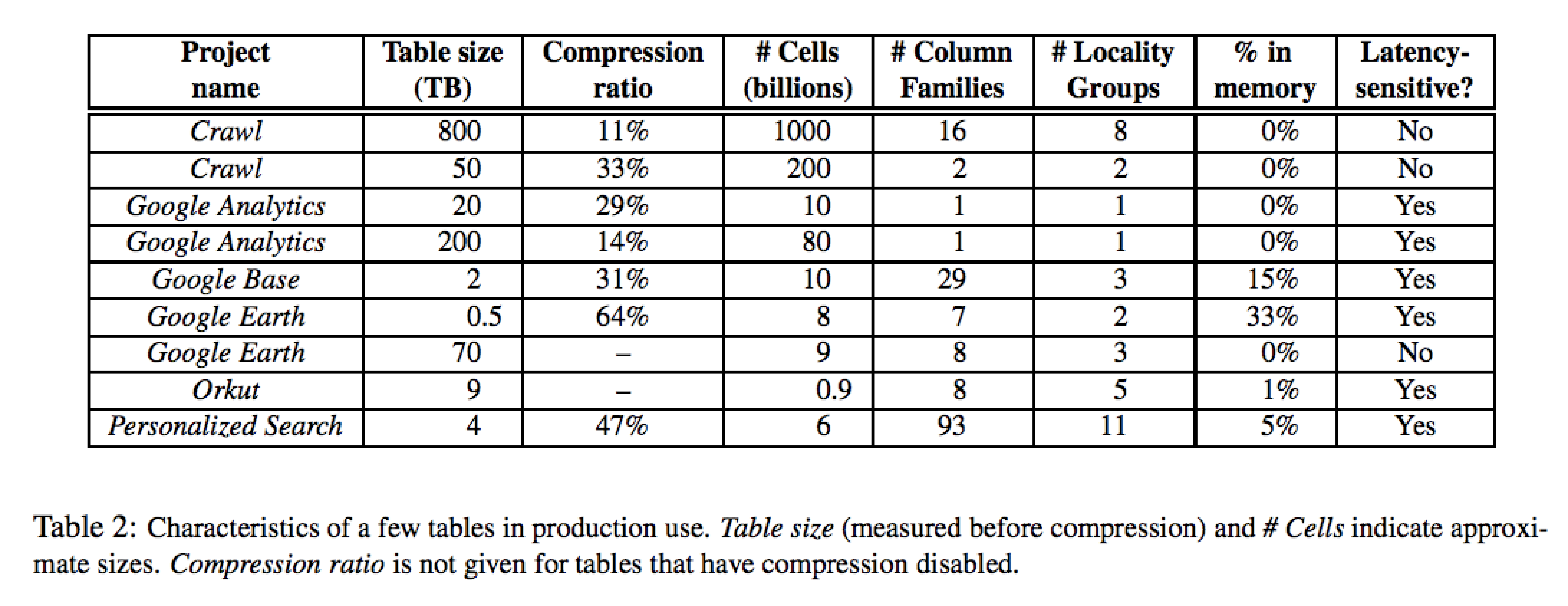
Real World BigTable

- 388 BigTable clusters in August 2006
- 24,500 tablet servers
- 14 clusters of 8069 servers served :
  - 1.2 M req/s
  - RPC inbound : 741MB/s
  - RPC outbound : 16GB/s

Google Analytics

- Raw click table (roughly 200TB)
- Summary table (20 TB)
- Summary generated from raw click with MapReduce

More stats



Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes

Table 2: Characteristics of a few tables in production use. *Table size* (measured before compression) and *# Cells* indicate approximate sizes. *Compression ratio* is not given for tables that have compression disabled.

Sawzall

- Query language
- Record-oriented
- Queries run as MapReduce on BigTable

Sawzall usage at google

- Over 2000 source files in the Google repository

About Erlang

- Concurrency oriented programming language
- No shared state
- Message passing only
- Functional core
- http://www.erlang.com

Erlang/OTP

- OTP : set of libraries for developping long running highly available servers
- also : *mnesia* distributed DBMS written in erlang

Mnesia

- Replicated failsafe database
- Not really relational
- Though it can be queried in a relational way
- Used to store arbitrary erlang terms

CouchDB

- Distributed non relational database
- Two way replication
- Access through a REST interface
- MapReduce runs queries written in JavaScript
- JSON is data format used in recent version

JSON

- JavaScript Object Notation

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
 }}
```

From Wikipedia

REST

- REpresentational State Transfer
- Resource oriented architecture (not SOA)
- Based on GET POST PUT DELETE
- For CRUD operations
- On URIs with a meaning

Public REST APIs

- Amazon Web Services : S3
- Flickr API
- Ziki API

CouchDB

- http://www.couchdb.org/
- Main developer : Damien Katz
  - Worked on Lotus Notes and MySQL
- Still in alpha, but very promising

Architecture

- Organized in Databases
- Databases store Documents
- Databases also store Views
- Views query Documents and present extracted data

Replicating

```
couch_rep:replicate
  ("database_name_a",
    "database_name_b").
```

Examples

- Admin interface available here :
- http://localhost:8888/_utils/browse/index.html
- Sample view :
-
```
function(doc) {
    if(doc.dogs.length > 0)
          return doc;
 }
```

Ruby code

- CouchDB Ruby bindings
- Not working very well ... yet (CouchDB api a bit hard to follow at the moment)

References

- Tim O’Reilly Radar : Database War Stories

  http://radar.oreilly.com/archives/2006/04/web_20_and_databases_part_1_se.html

- Google papers (MapReduce, GFS, BigTable, Sawzall)

  http://research.google.com/archive/mapreduce.html
  http://labs.google.com/papers/gfs.html
  http://labs.google.com/papers/bigtable.html
  http://labs.google.com/papers/sawzall.html

- CouchDB installation

  http://intertwingly.net/blog/2007/09/04/Building-CouchDB

- Erlang

  Programming Erlang, Joe Armstrong, Pragmatic Programmers 2007