Lecture 1 – Overview and Perspectives

922EU3870 – Cloud Computing and Mobile Platforms, Autumn 2009

2009/9/14

Ping Yeh (葉平), Google, Inc.

The Morakot story

Morakot

- 2000mm of rain poured in southern Taiwan in 1.5 days during holidays.
- Flood, mudslide.
- People crying for help, emergency phone line is jammed.
- 3 web sites were created for information exchange.
  - typhoon.adct.org.tw, typhoon.oooo.tw, disastertw.com.
- Other people used existing tools to create web pages.
  - 莫拉克颱風災情地圖, PTT Emergency 網頁版, PTT資訊連結整合(含志工), 救援物資集散地
- Other people post to microblogging sites: twitter, plurk.

typhoon.adct.org.tw

disastertw.com

莫拉克颱風災情支援網

Google™ 自訂搜尋 搜尋

首頁 地區災情 [刊登資訊 / 發出求救] 捐贈物資資訊 捐款、募款資訊 !! 志工招募 !! 站內搜尋 連絡站長

求救 尋人 提供物資 需要物資 名單 / 公告 快報 其他 純報平安 志工資訊 [刊登資訊 / 發出求救]

請求支援列表

[推到 Plurk!] | （若有來亂的請不吝檢舉） | 消防署已將本站提供給各縣市災害應變中心救災參考(來源) | !! 按此刷新 !! | 莫拉克災情資料表 (另站)

« Previous 1 2 3 4 5 6 7 8 9 ... 12 13 Next »

類型	縣市	主旨	內容	最新狀況	Created at
快報	高雄縣	明天桃園幾個村大撤離	明天桃園村，建山村，梅蘭村，高中村要撤村，搜救人員、國軍等各種支援人員都要撤離。 以上是確定的訊息 ---------- 以下是我個人的看法 請桃園自救會要花一些心思確定一下，是不是有一些散居的住戶沒有被撤離到，如果沒撤到，真的就是放在那邊自生自滅了，這次撤了之後可能就要等到道路搶通了，對這幾個村應該不會再空投了吧。	Update	2009/08/15 12:39 AM | [檢舉]

typhoon.oooo.tw

序號	時間	鄉鎮市	詳細地址	連絡方式	發生災情	需要物資、協助	目前最新狀態
1706	08/11 18:35	高雄縣那瑪夏鄉	高雄縣那瑪夏鄉民生村	黃先生 0910-021887 或 0980-937709	電話可通,可是都沒有人接,我擔心我家人的安危,請給我有辦法聯落進去的電話,我家人 周孔義.周慶恩.請給我可連絡進去的辦法,謝謝各位.	日常用品,食物.	新增 [08/11 18:47] 周慶恩是否與周雪婷同一家人？我在十年前於民生村與他們有一面之緣，願主保守他們一家人平安！ [08/11 18:52] 是的!他們同一家!請問有他們的消息嗎? [08/12 21:41] 願上帝保佑他們以及三民鄉所有的鄉民...香菇寮翁偉剛.....
1704	08/11 18:33	阿里山奮起湖	阿里山奮起湖	0910199944	尋奮起湖替代役藍金寶，有看到消息請儘速聯絡。目前聯絡不到你媽媽，她有可能在寶來溫泉附近。		新增
1703	08/11 18:29	高雄縣甲仙鄉與六龜鄉	高雄縣甲仙鄉與六龜鄉	0910669550 顏小姐	甲仙與六龜兩鄉鎮災害慘重,急需人員前往支援救災	急需徵求有經驗的救難志工前往災區參與救難任務	新增 [08/11 22:02] 那邊目前有路可以到新發、舊潭、興龍嗎？！ 如有詳細路程 麻煩請告知！ 拜託！！

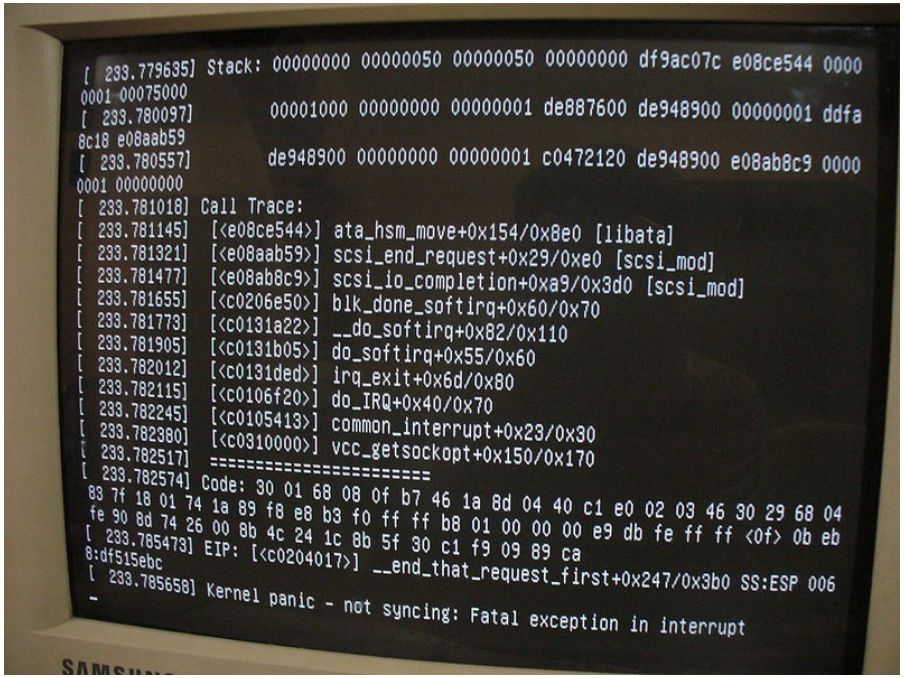
Problems!

- Two sites went down soon after people flock there.
  - One site spent hours to move to a telecom's data center.
  - The other went up and down several times.
- One site stayed functional all the time.

Why?

Load

- A server can handle limited load
  - Each request requires resources to handle: process, thread, CPU, memory, etc.
  - There is a maximum load one server can handle
- The database can handle limited connections
- Beyond the maximum:
  - can't accept new requests: timeout, intermittent service, no more connections, etc.
  - program crash
  - kernel panic

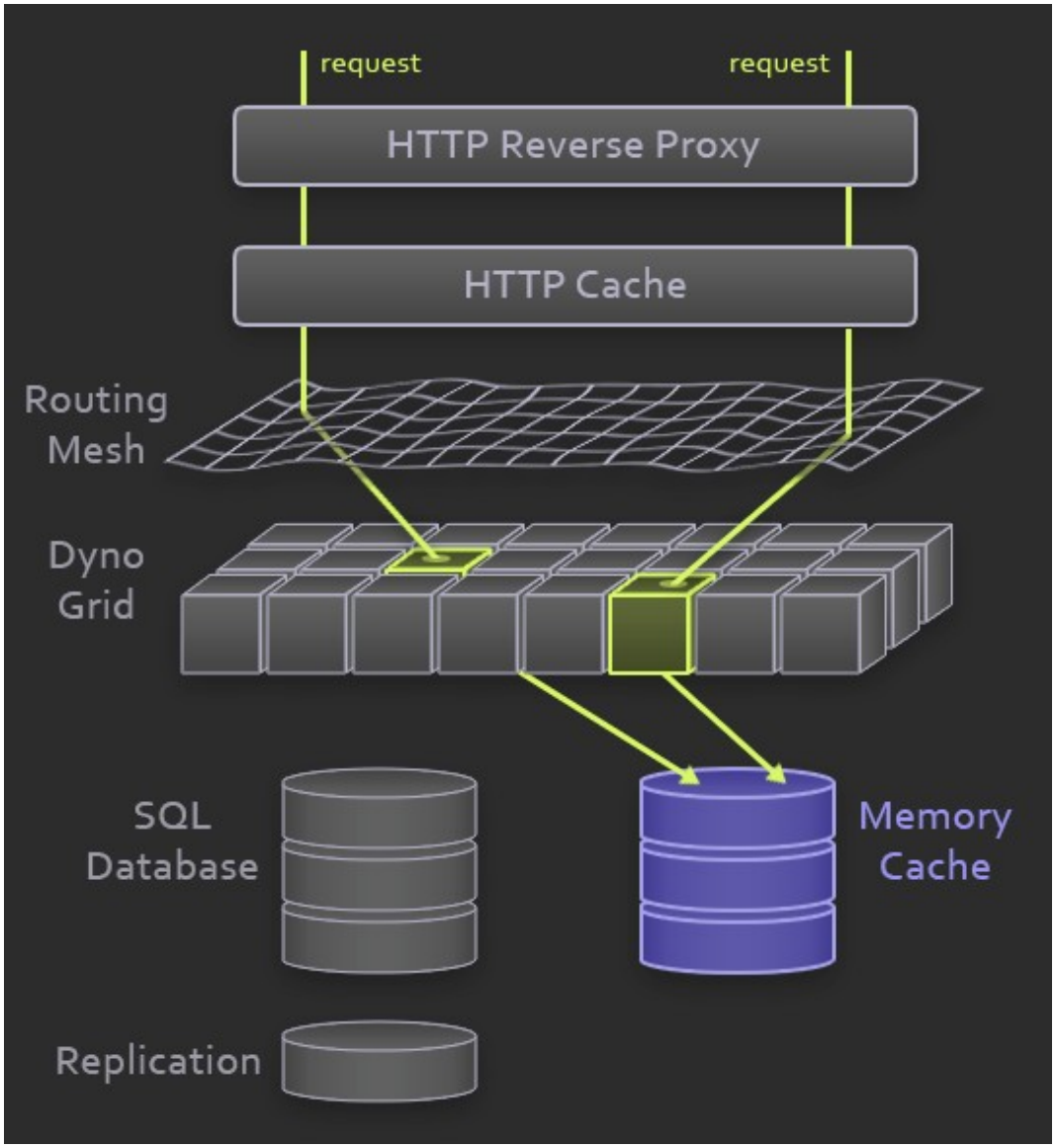


http://en.wikipedia.org/wiki/File:Ubuntu-linux-kernel-panic-by-jpangamarca.JPG

How did one site survive?

- One keyword: Scalability
- Key points on scalability:
  - Simple web site (not complicated CMS): increases QPS
  - Deployed on Heroku: up to 3 dynos for 400k pageviews/day
    - peak qps estimated: > 100.
    - quote http://heroku.com/how/dynos: “4 dynos are equivalent to the compute power of one CPU-core on other systems.”
  - Use content delivery network (CDN) for static data
  - Pay as you go: about US$62 for heroku and CDN combined.
- The developer's blog: http://blog.xdite.net/?p=1369

Heroku: http://heroku.com/



- A fast deployment environment for Ruby on Rails.
- Quote: “new dynos can be launched in under 2 seconds for most apps.”
- Built on top of Amazon's EC2 service.

What would people do if Morakot hit us in 1989? 1999?

1989

- Internet in Taiwan was in the labs of III and ITRI
- E-mails were mostly sent with Bitnet, only limited amount of universities have access.
- The only way to spread “call for help” and “looking for loved one” messages would be via telephones and newspapers.

1999

- Web was popular, about 4.8 million people were connected.
- Internet was prevalent in all universities, BBS was the dominant use.
- Portals were the center of attention on the web.
- The best way to spread “call for help” and “looking for loved one” messages would be through BBS.
  - requires manual data organization
  - could not handle too many connections at the same time

Since ~1999: “Web 2.0”

An old buzzword, means different thing to different people

- Self publishing: LiveJournal/Blogger.com (1999), YouTube (2005)
- Metadata & feeds: RSS (1999), Atom (2003)
- Collaboration: Wikpedia (2001), Google docs (2006)
- Tagging: del.icio.us (2003), digg.com (2004), flickr (2004)
- Ajax: Gmail (2004), Google Maps (2005)
- Long Tail: Amazon (1994), Google AdSense (2005)
- API: Yahoo!, GData, Amazon… (who doesn't?)
- Aggregation: Google News (2002), Google Reader (2005)
- Social: LinkedIn (2003), MySpace (2003), Facebook (2004)
- Microblogging: Twitter (2006), Plurk (2008)

No matter if you call it “Web 2.0,”

you can't deny that

more people are spending more

time on the web.

Web sites are becoming applications

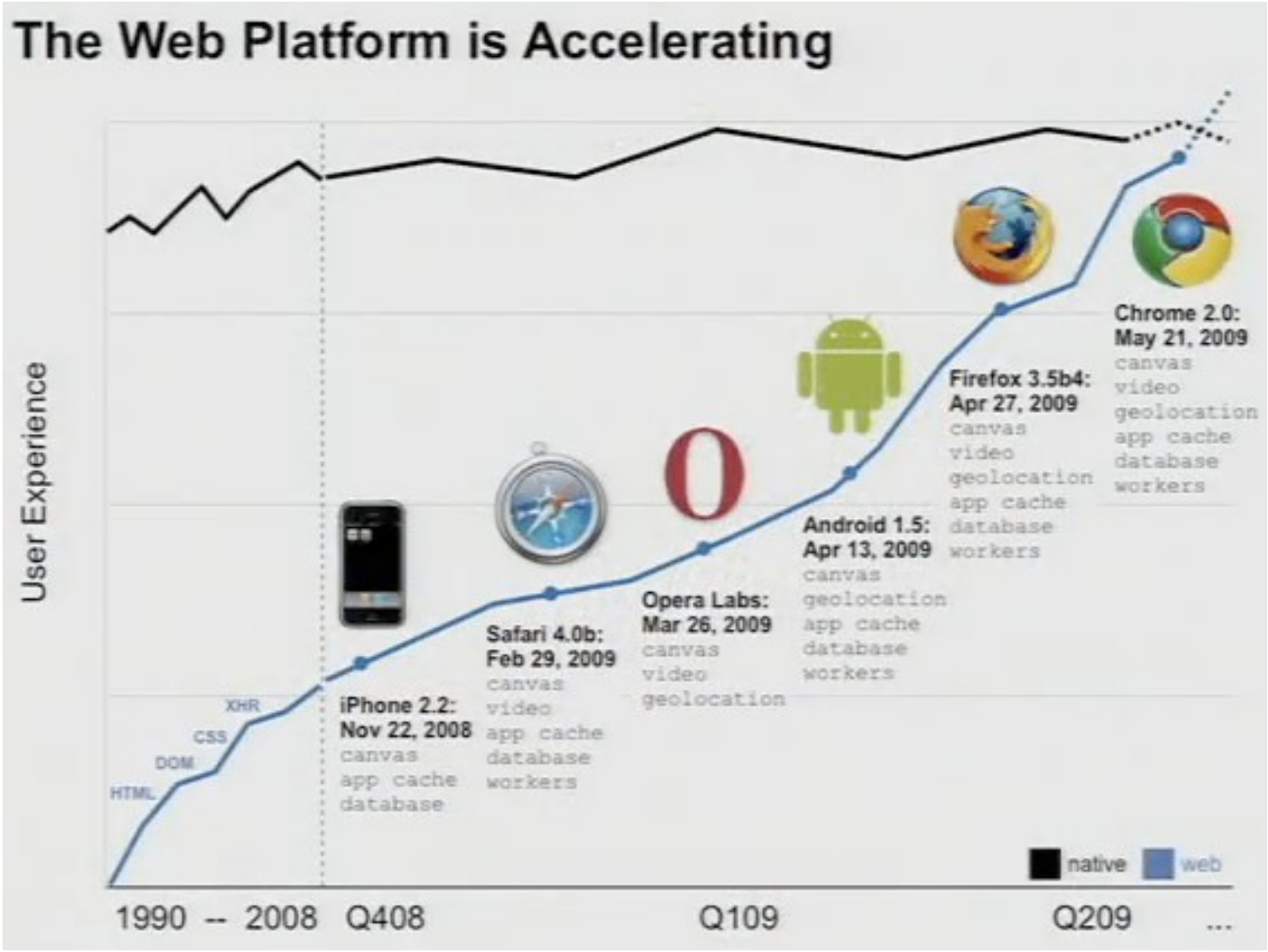
- Search: ease of finding information
- Tags: making things easier to find without hierarchy
- Authoring: ease of writing things
- Rich interaction: intuitive and streamlined interface
- Aggregation: data, application
- Social: discover friends, stay in contact, organize events

And developers/users can mix them up!

5 > 2.0

- Emerging new technologies: HTML5
  - canvas tag (demo at http://htmlfive.appspot.com/)
  - video tag (demo)
  - Geolocation API (demo)
  - App caching and databases
  - Workers (demo)
  - Document editing, drag and drop, browser history management, ...
- Other new stuff:
  - 3D (demo at 23:20 of Google IO Keynote)
- (Re)newed stuff:
  - Native SVG support

The Web as a Platform

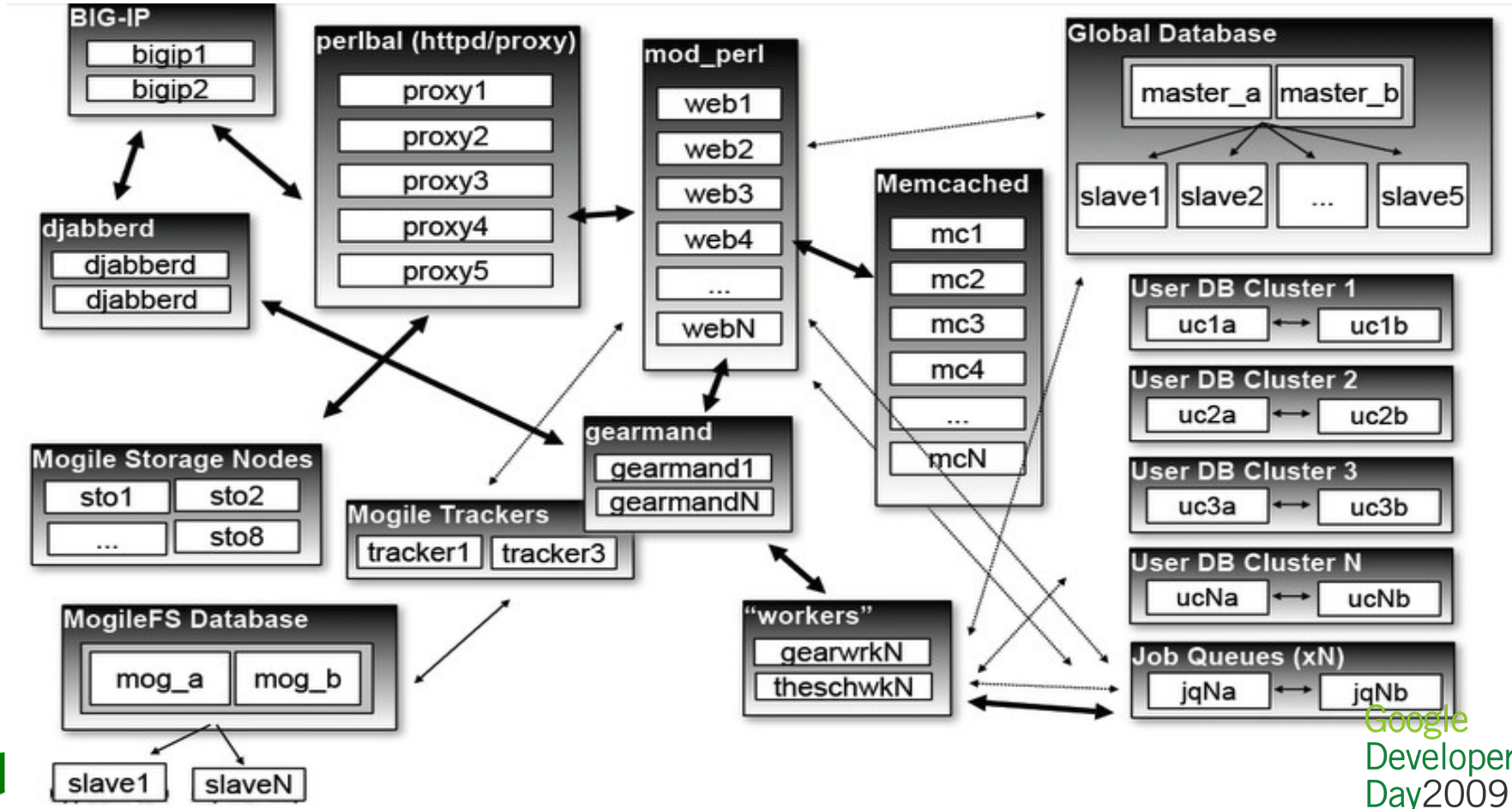


Lessons Learned

- Having a group of capable web developers is not enough for building a successful web site.
- Scalability is key, because "Not serving = Useless"
  - Reverse proxy
  - HTTP cache
  - Database
  - Memory cache
  - Alternative delivery mechanism for static contents
  - etc
- Sometimes using existing web tools is better – no need to recreate the wheels.

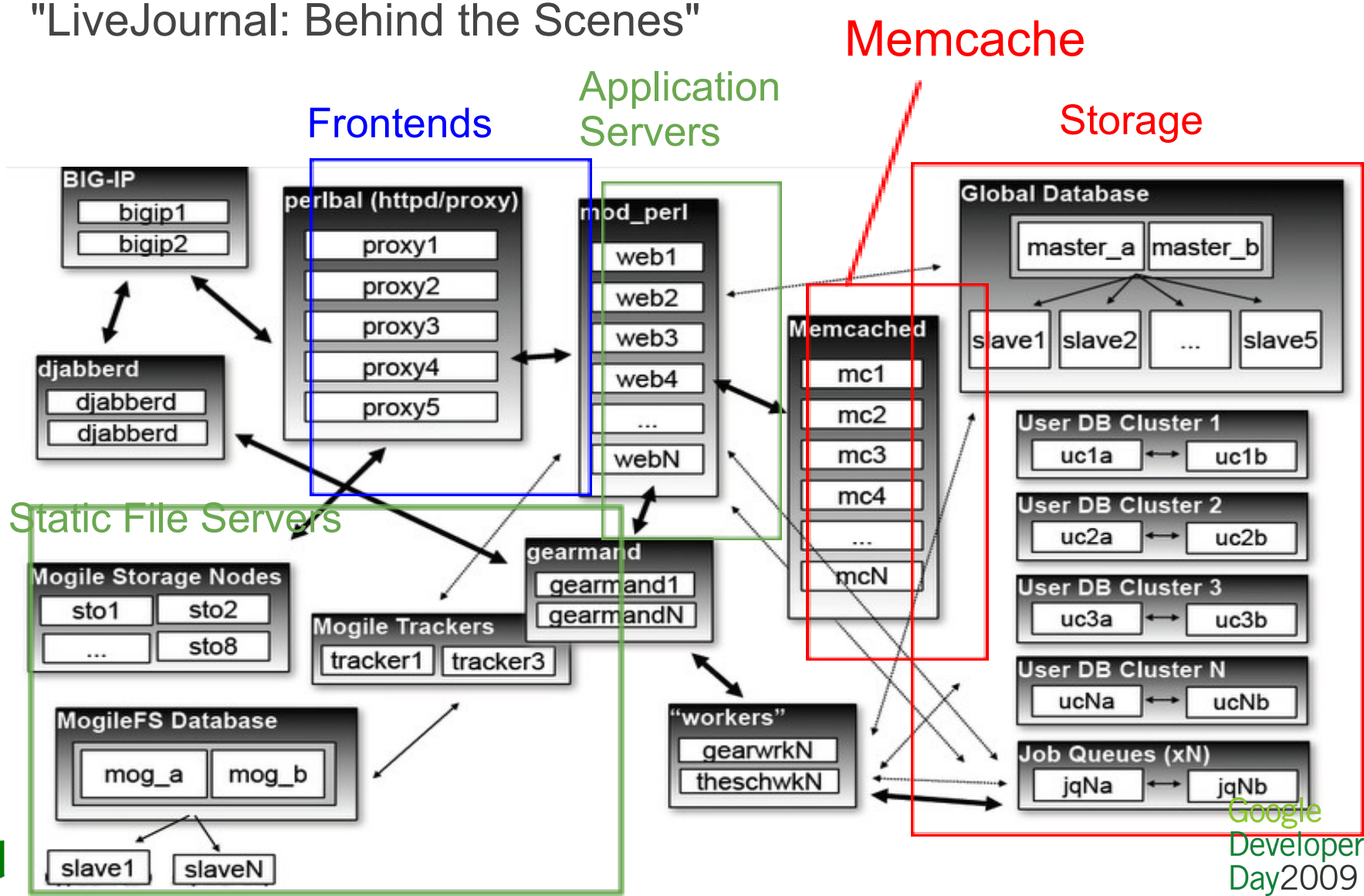
LiveJournal circa 2007

From Brad Fitzpatrick's USENIX '07 talk:
"LiveJournal: Behind the Scenes"



LiveJournal circa 2007

From Brad Fitzpatrick's USENIX '07 talk:
"LiveJournal: Behind the Scenes"



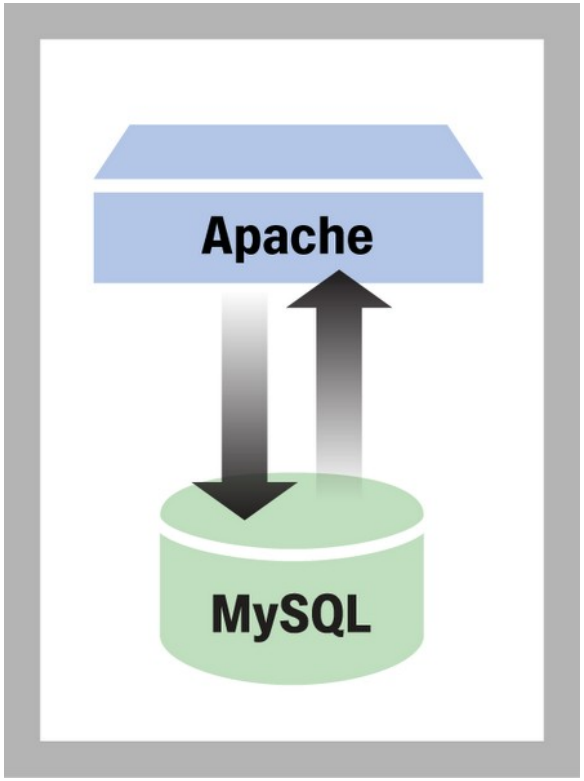
Basic LAMP

LAMP

Linux + Apache + MySQL +

Programming Language

(perl, Python, PHP, ...)



Scalable?

- Shared machine for database and webserver

Reliable?

- Single point of failure (SPOF)

Dedicated Database

Database running on a separate server

Requirements:

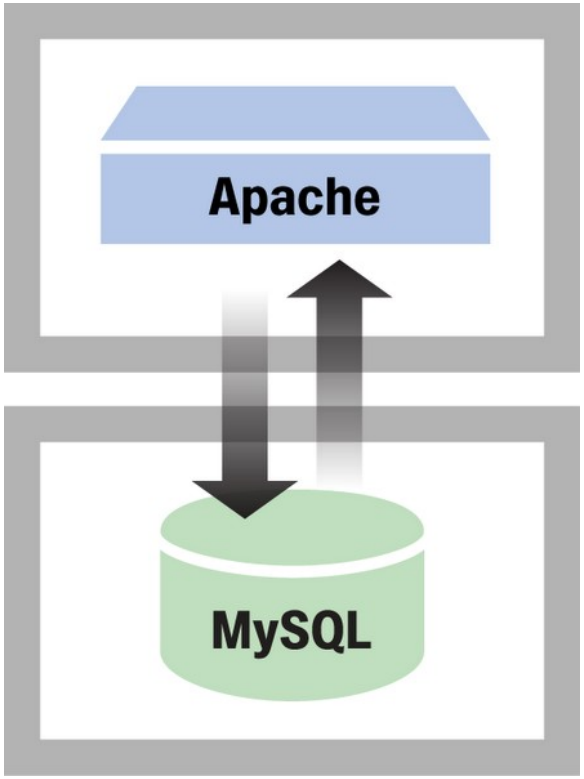
- Another machine plus additional management

Scalable?

- Up to one web server

Reliable?

- Two single points of failure



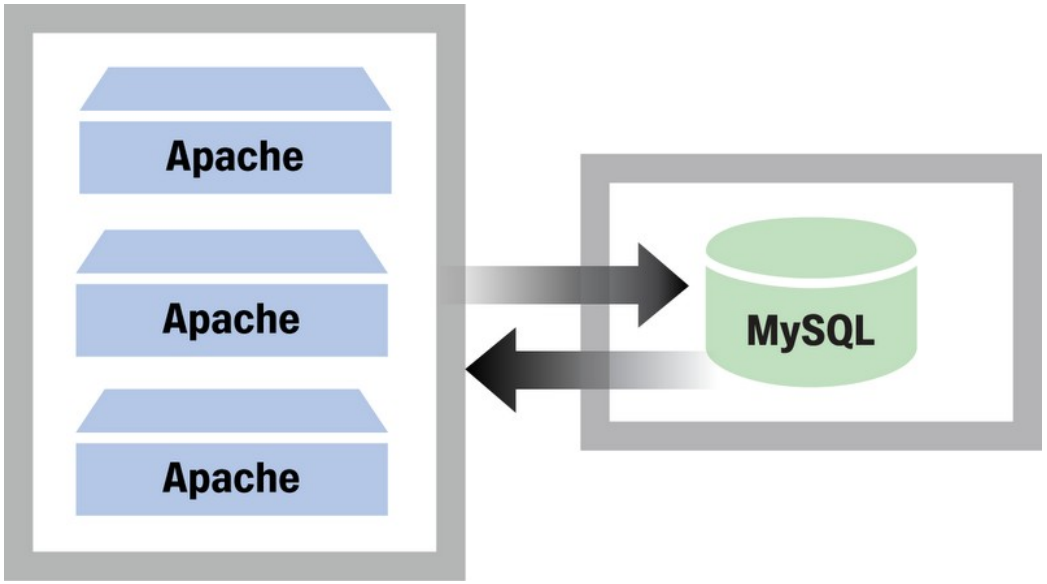
Multiple Web Servers

Benefits:

- Grow traffic beyond the capacity of one webserver

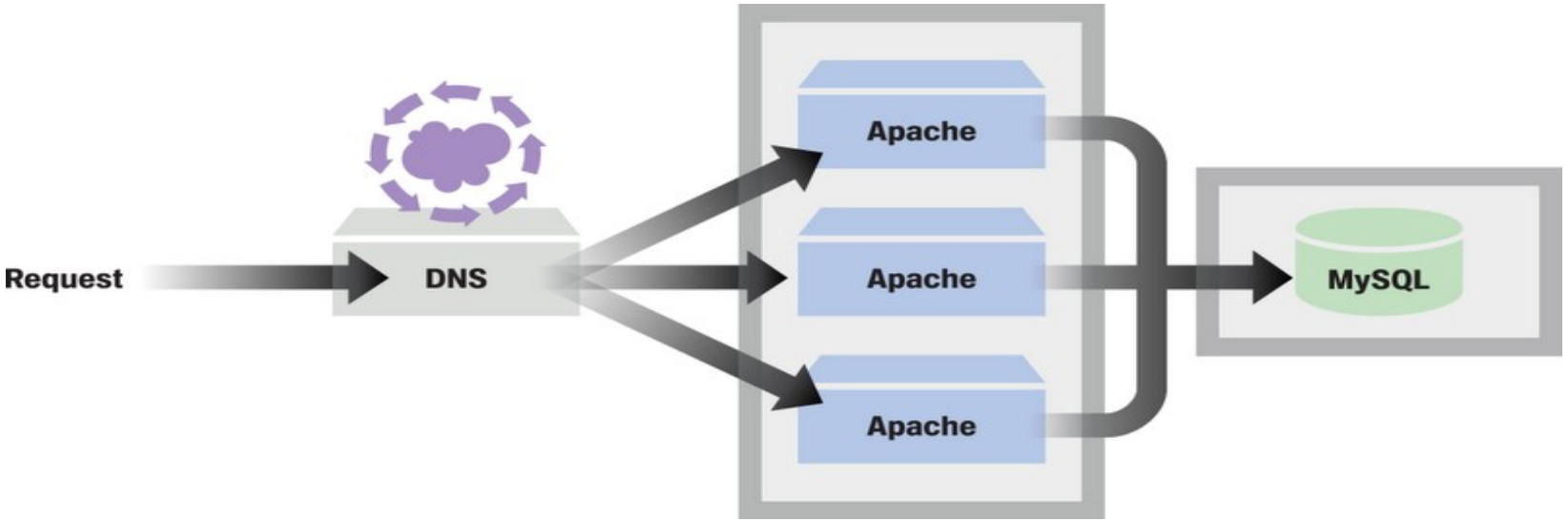
Requirements:

- More machines
- Set up load balancing



Load Balance: DNS Round Robin

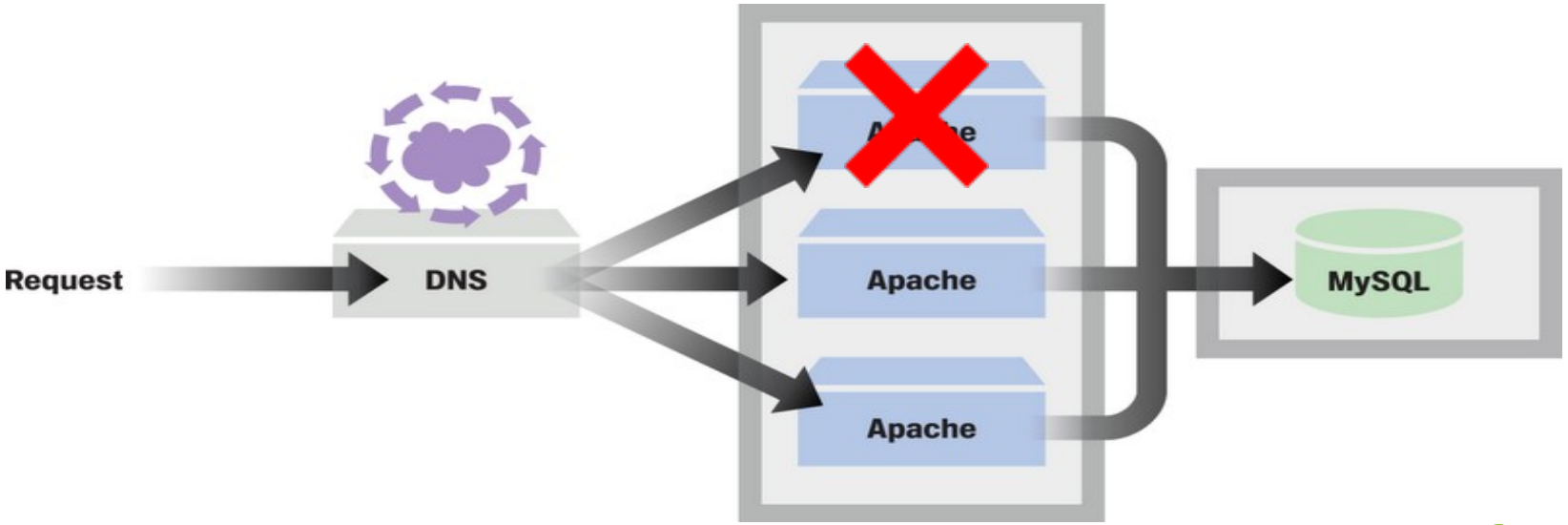
- Register list of IPs with DNS
- Statistical load balancing
- DNS record is cached with a **Time To Live** (TTL)
  - TTL may not be respected



Load Balance: DNS Round Robin

- Register list of IPs with DNS
- Statistical load balancing
- DNS record is cached with a **Time To Live** (TTL)
  - TTL may not be respected

Now wait for DNS changes to propagate :-(



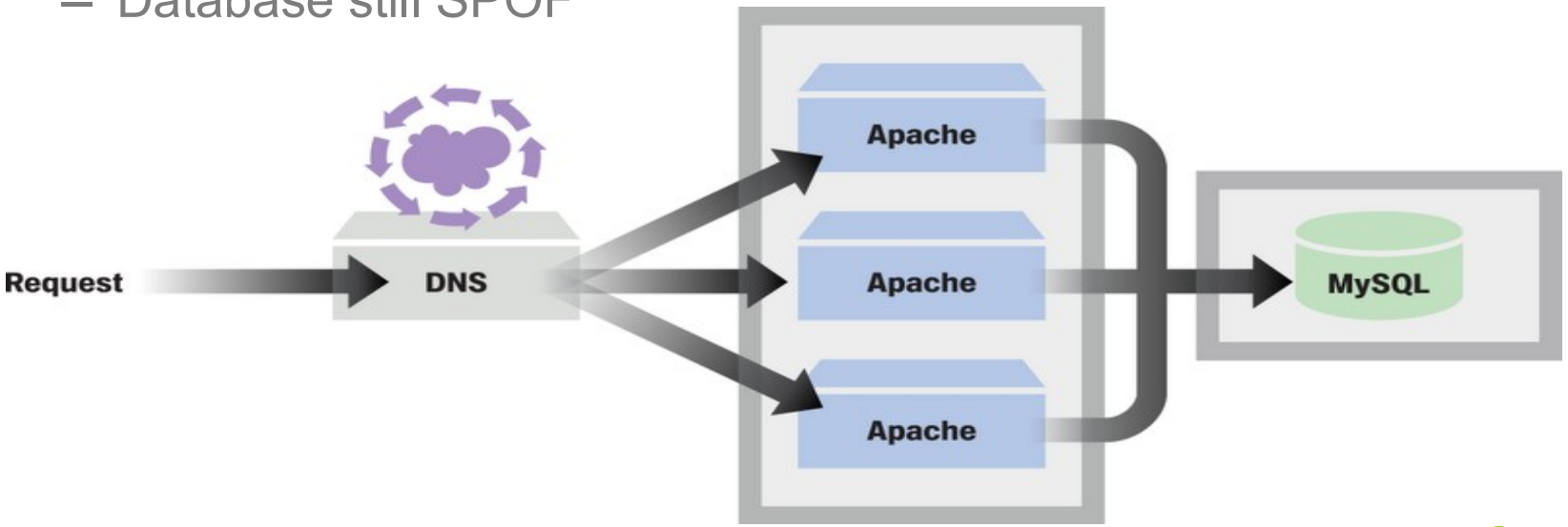
Load Balance: DNS Round Robin

Scalable?

- Add more webservers as necessary
- Still I/O bound on one database

Reliable?

- Cannot redirect traffic quickly
- Database still SPOF



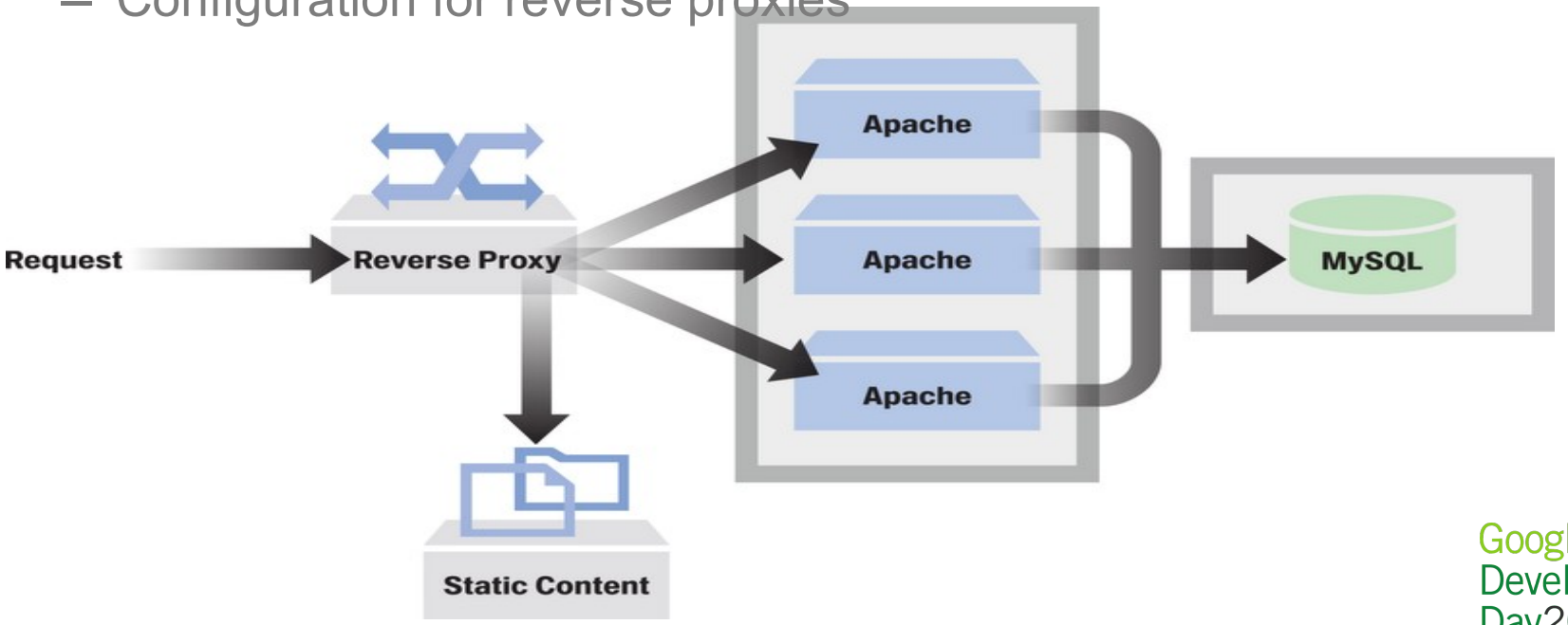
Reverse Proxy

Benefits: Custom Routing

- Specialization
- Application-level load balancing

Requirements:

- More machines
- Configuration for reverse proxies



Reverse Proxy

Scalable?

- Add more web servers
- Bound by
  - Routing capacity of reverse proxy
  - One database server

Reliable?

- Agile application-level routing
- Specialized components are more robust
- Multiple reverse proxies requires network-level routing
  - Fancy network routing hardware
- Database is still SPOF

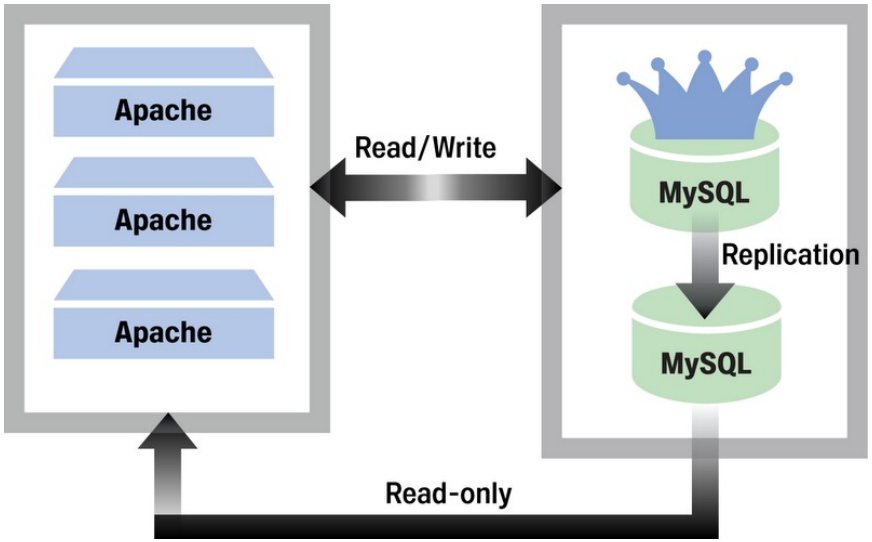
Master-Slave Database

Benefits:

- Better read throughput
- Invisible to application

Requirements:

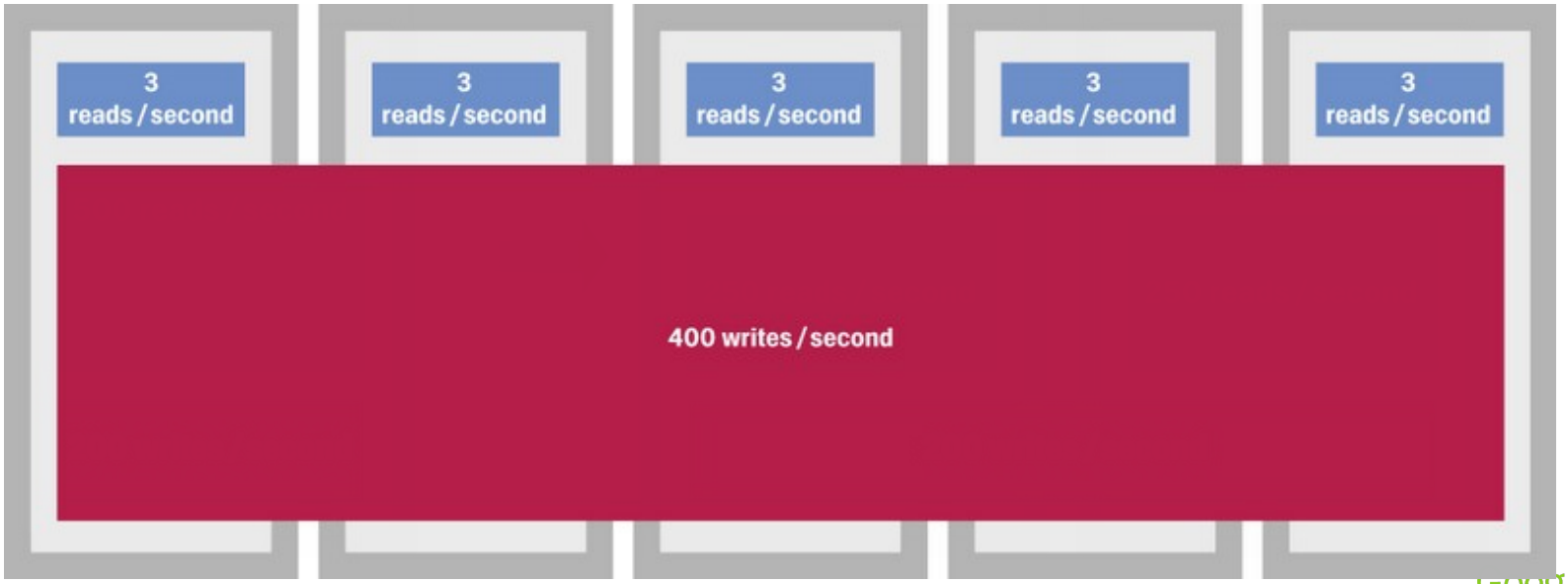
- Even more machines
- Changes to MySQL, additional maintenance



Master-Slave Database

Scalable?

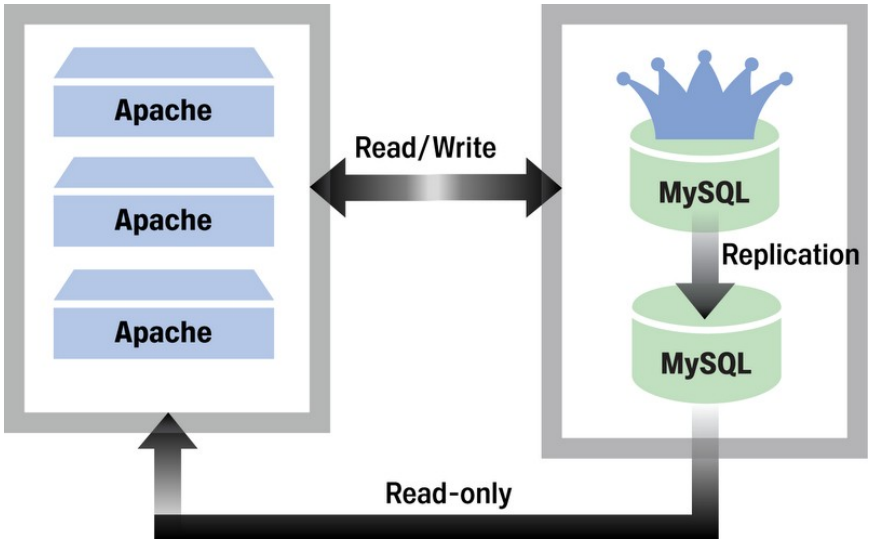
- Scales read rate with # of servers
  - But not writes
- But what happens eventually?



Master-Slave Database

Reliable?

- Master is SPOF for writes
- Master may die before replication



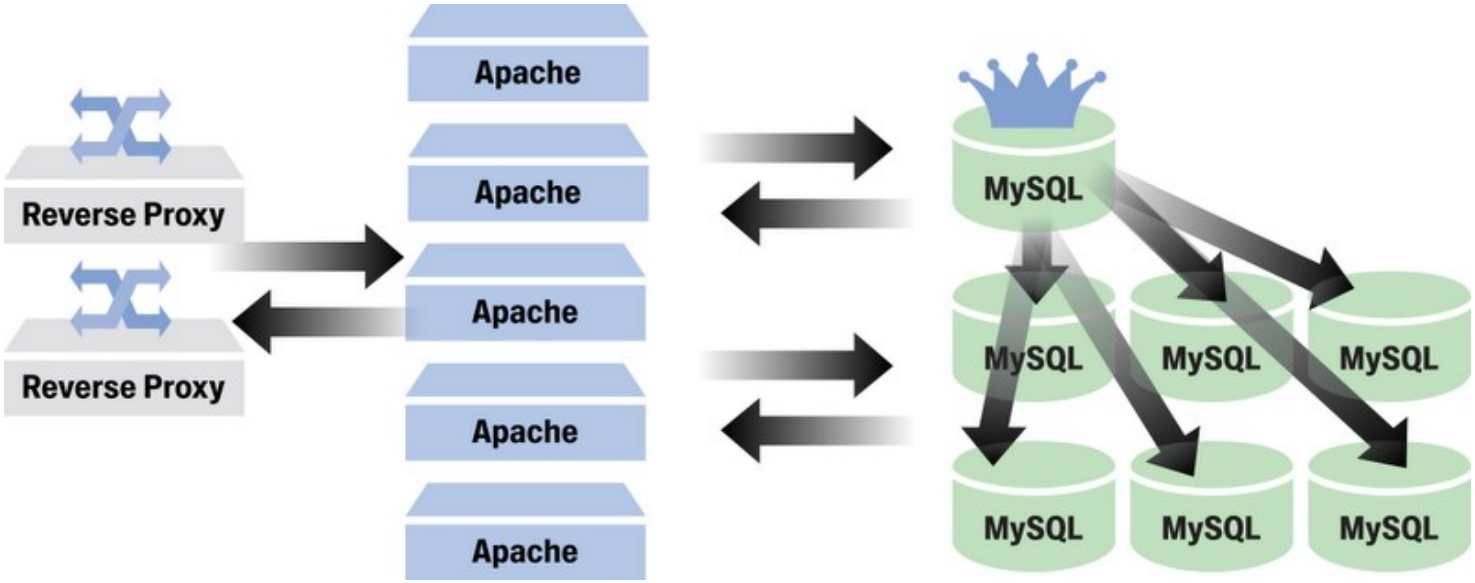
Partitioned Database

Benefits:

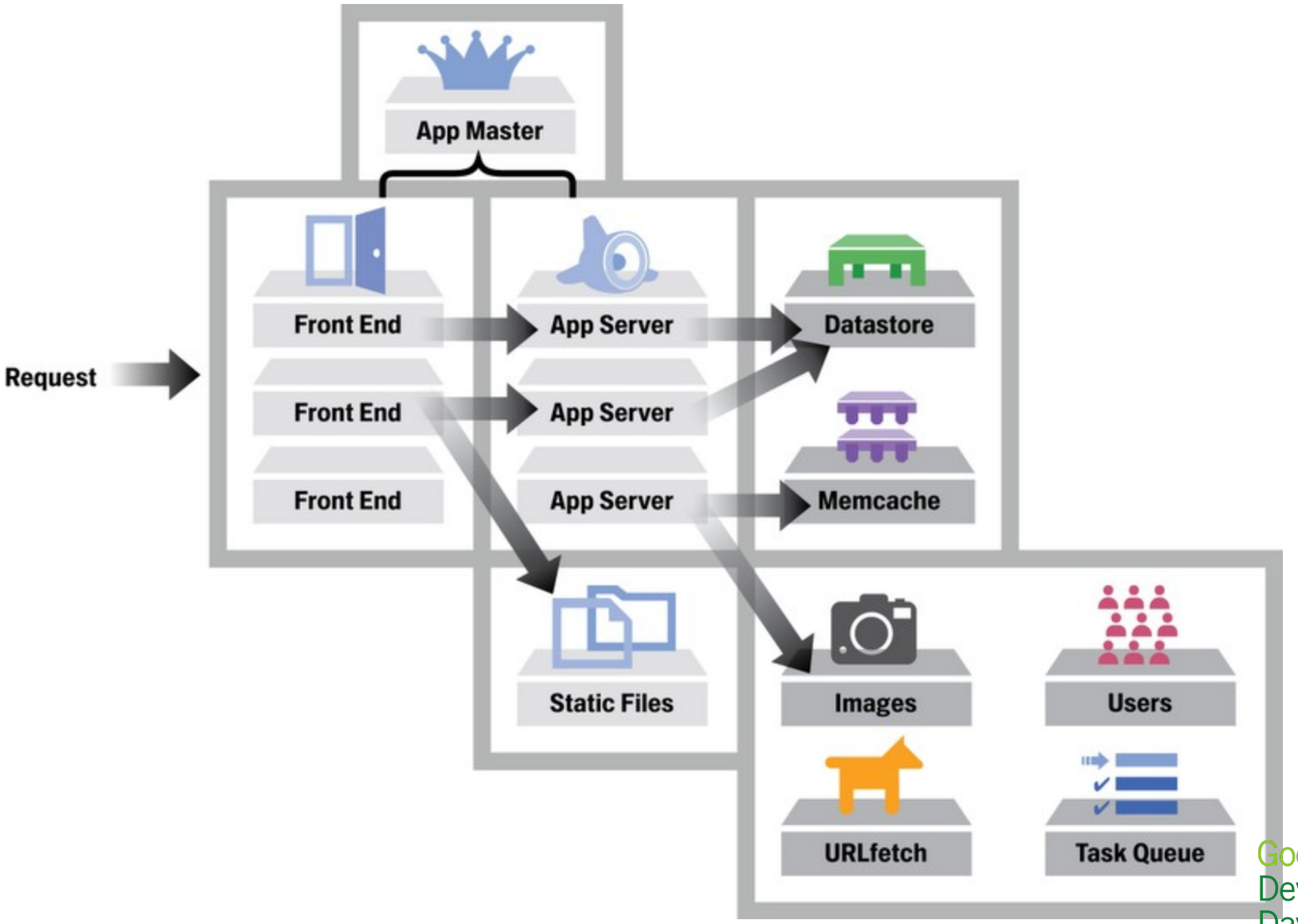
- Increase in both read and write throughput

Requirements:

- Even more machines
- Lots of management
- Re-architect data model



How about Google App Engine?



How do people describe Cloud Computing in public?

The International Workshop on Cloud Computing - 2009

http://www.icumt.org/w-35.html

Cloud Computing is defined as a pool of virtualized computer resources. Based on this virtualization the Cloud Computing paradigm allows workloads to be deployed and scaled-out quickly through the rapid provisioning of virtual machines. A Cloud Computing platform supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many inevitable hardware/software failures and monitoring resource use in real time for providing physical and virtual servers on which the applications can run. A Cloud Computing platform is more than a collection of computer resources because it provides a mechanism to manage those resources. In a Cloud Computing platform software is migrating from the desktop into the "clouds" of the Internet, promising users anytime, anywhere access to their programs and data. What challenges does it present to users, programmers, and telecommunications providers?

Wikipedia

http://en.wikipedia.org/wiki/Cloud_computing

Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet.[1][2] Users need not have knowledge of, expertise in, or control over the technology infrastructure in the "cloud" that supports them.[3]

The concept generally incorporates combinations of the following:

- infrastructure as a service (IaaS)
- platform as a service (PaaS)
- software as a service (SaaS)
- Other recent (ca. 2007–09)[4][5] technologies that rely on the Internet to satisfy the computing needs of users.

SearchCloudComputing.com

http://searchcloudcomputing.com/sDefinition/0,,sid201_gci1287881,00.html

Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flow charts and diagrams.

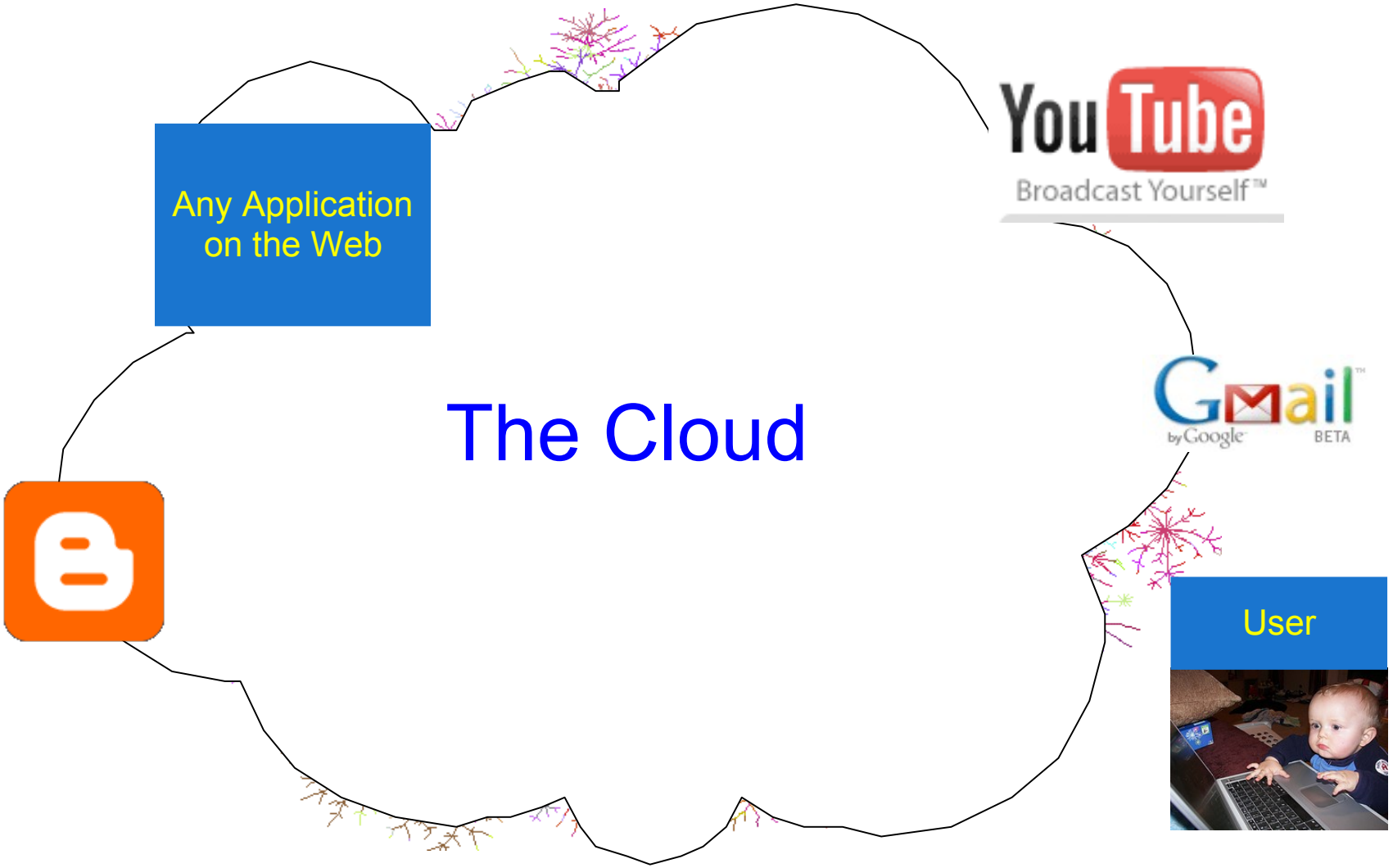
A cloud service has three distinct characteristics that differentiate it from traditional hosting. It is sold on demand, typically by the minute or the hour; it is elastic -- a user can have as much or as little of a service as they want at any given time; and the service is fully managed by the provider (the consumer needs nothing but a personal computer and Internet access). Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet and a weak economy, have accelerated interest in cloud computing.

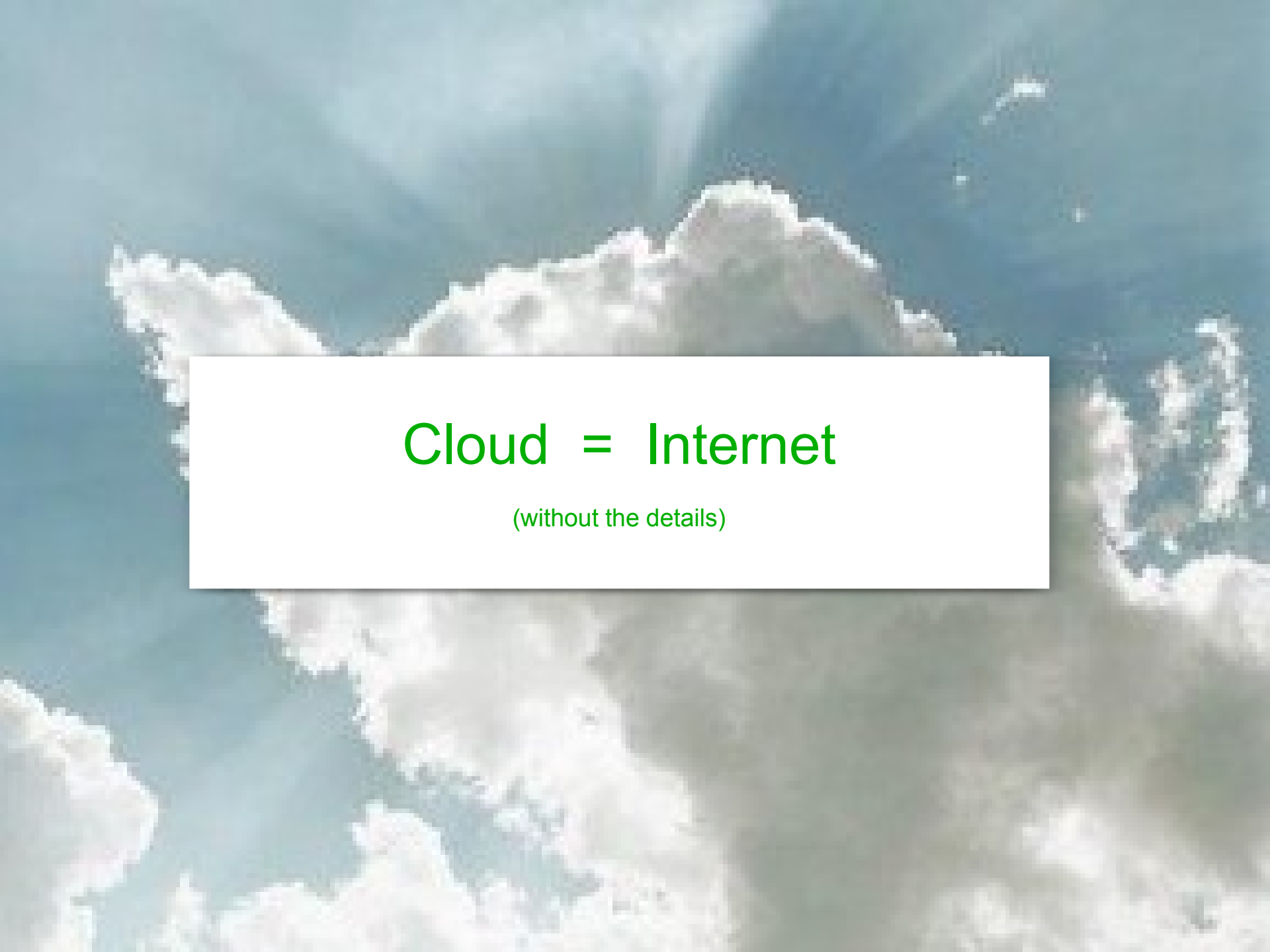
Information Weeks' John Foley

http://www.informationweek.com/cloud-computing/blog/archives/2008/09/a_definition_of.html

Cloud computing is on-demand access to virtualized IT resources that are housed outside of your own data center, shared by others, simple to use, paid for via subscription, and accessed over the Web.

What is cloud?




Cloud = Internet
(without the details)

Cloud Computing:

Applications and data are in the cloud,
accessible with any device
connected to the cloud
with a browser

Cloud Computing...

- is about the paradigm
  - web
  - application, data
  - anyone with any device with a browser
- focuses on internet scale
  - easy to use
  - elastic, on-demand
- is less about the implementation
  - virtualization
  - outside your own data center
  - shared, subscription

Three Types of Cloud Services

- Software as a Service: Any web site that handles large scale of users and computing. Examples: Google search, Amazon, Facebook, Twitter, Salesforce, etc.
- Infrastructure as a Service: Computing resources that provide elastic on-demand resource for rent. Examples: Amazon EC2, GoGrid, 3tera, etc.
- Platform as a Service: Software development and deployment environment on top of a Cloud infrastructure. Example: Google App Engine, Heroku, Aptana, etc.

Evolution of Computing with the Network

Network Computing

Separation of Functionalities

Cluster Computing

Commodity, Open Source

Grid Computing

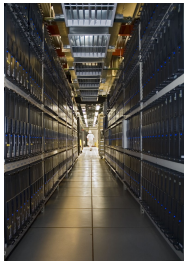
Global Resource Sharing

Utility Computing

Ownership Model

Network is computer (client - server)

Tightly coupled computing resources:

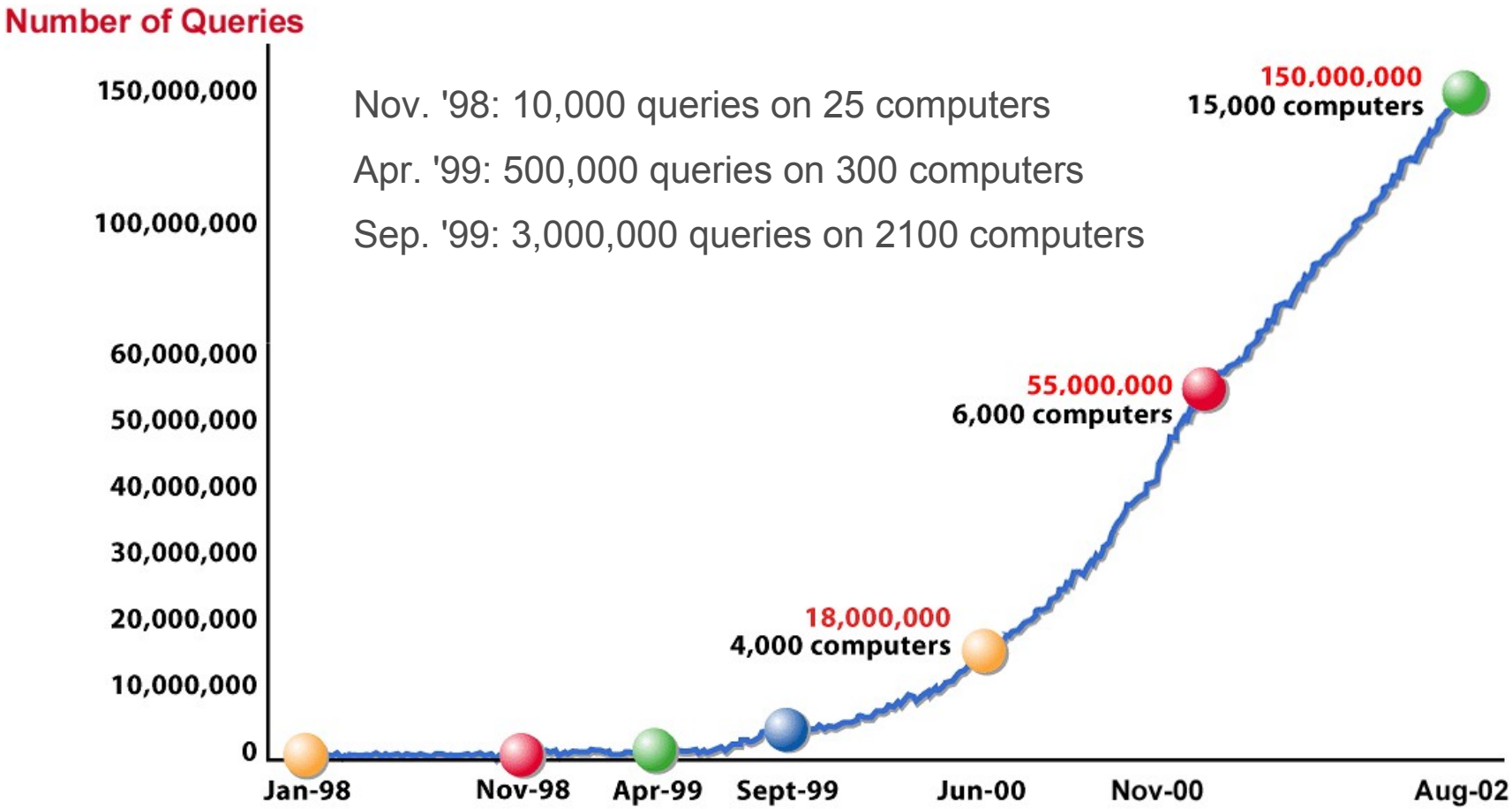


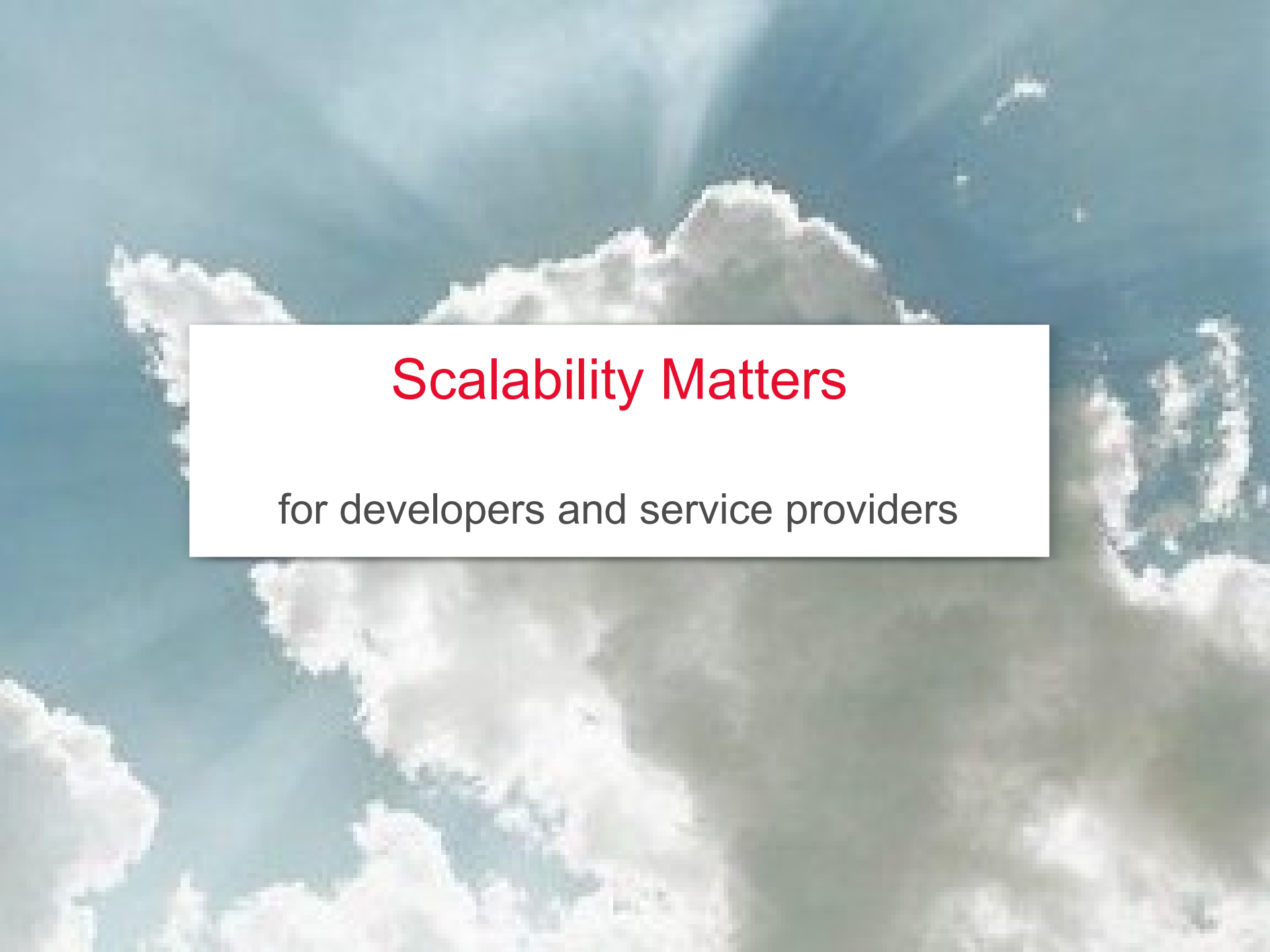
Don't buy computers, lease computing power
Upload, run, download

Cluster and grid images are from Fermilab and CERN, respectively.

How many users do you want to serve?

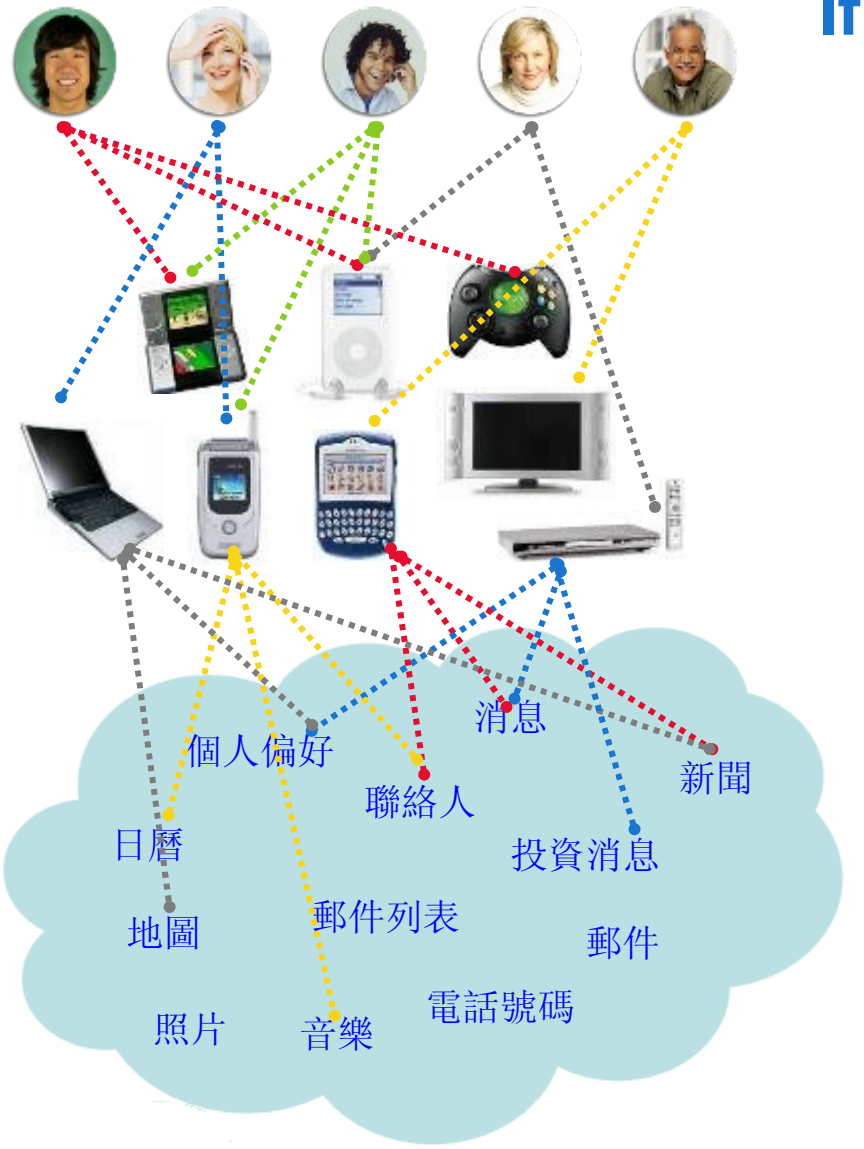
Google Growth




Scalability Matters
for developers and service providers

What does Cloud Computing mean to ordinary users?

User-centric



If your data are in the cloud...

It "follows" you

Only need a browser, username and password to get your data.

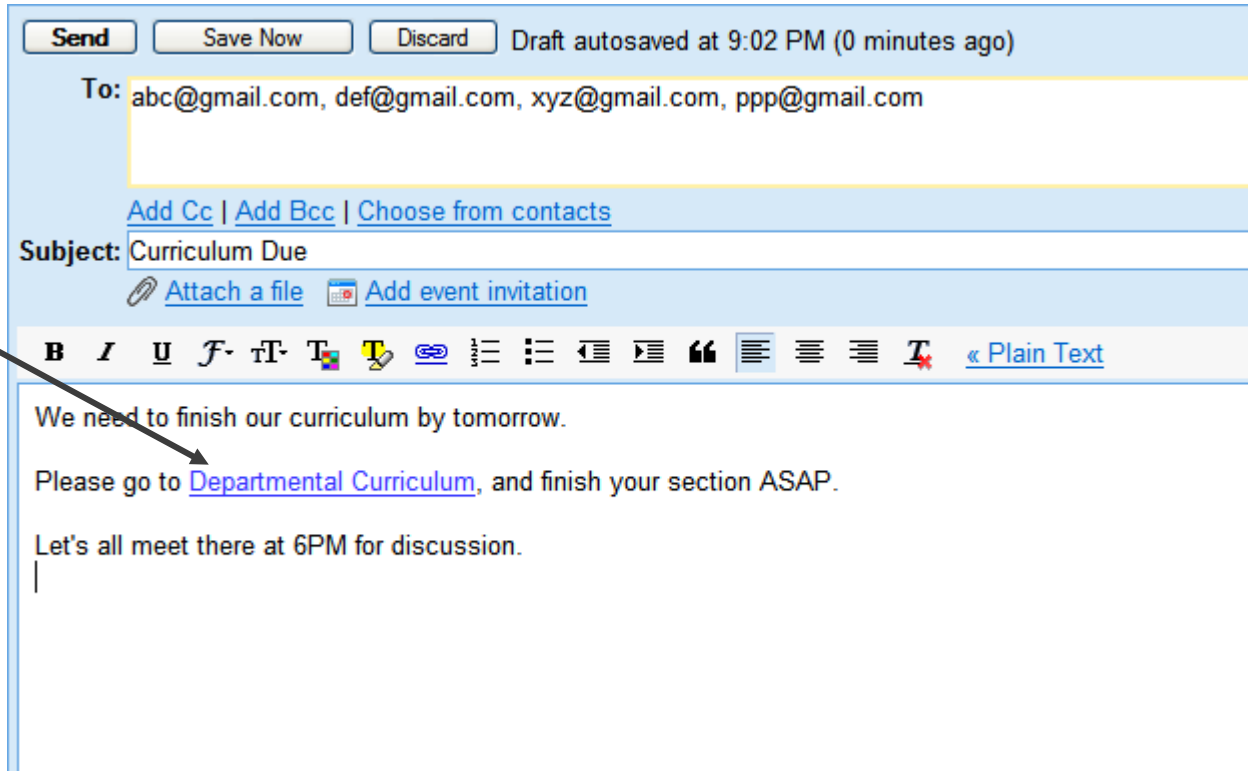
Once logged in, the device becomes "yours"

You can get your data when you want it, where you want it.

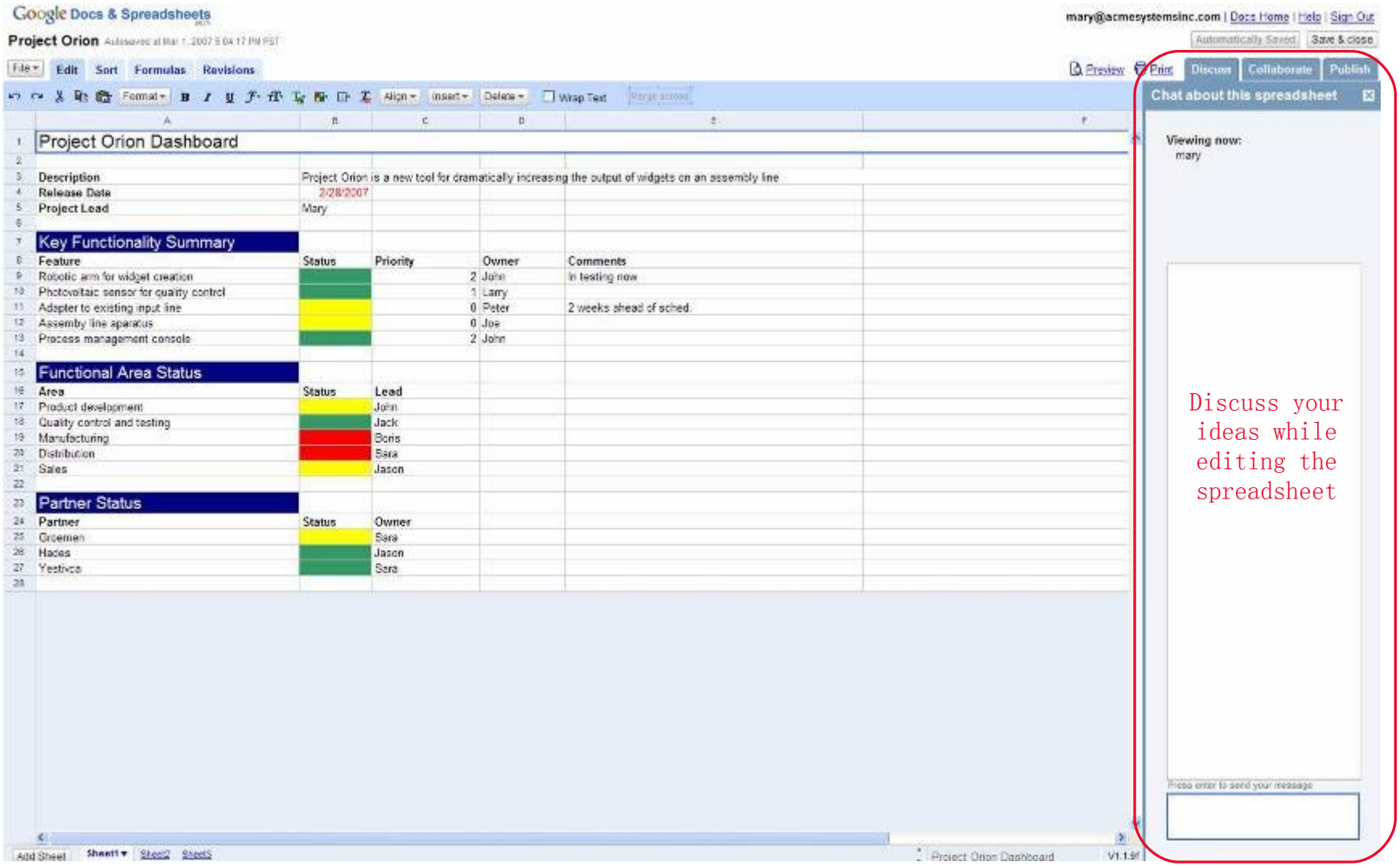
Easy to share, controlled by you.

Everything has a URL

URL is a presence of the task - a central point for related data and application in the cloud.



Collaboration on the web



Intelligent and Powerful

Example: Search

vs.

Your typical desktop document application

Speed	~ 0.25s	~ 1s
Relevance	high	low
Data size	O(100TB)	O(1MB)

What does Cloud Computing mean to web application developers?

Understand Internet Scale Computing

- Serving: dynamic content vs. static content, caching, latency, load balancing, etc.
- Database: partitioned data (?), replication, data model, memory cache, etc.
- Offline processing: how to process internet-scale of input data to produce interesting output data for every user?

- Don't build all the wheels! Use what's available.
  - If a PaaS supports your favorite language, check it out.
  - Else, check out the IaaS providers for your web app.
  - Hosting your web app on your own servers at home or office: not a wise move unless you predict there will not be many users.

Cloud Software

- Google: GFS, BigTable, MapReduce. Not directly available publicly.
- Hadoop: open source implementation of GFS and MapReduce in Java.
- HBase, Hypertable: open source implementation of BigTable in Java, C++.
- Thrift: open source RPC framework by Facebook.
- Protocol Buffers: open source data serialization and RPC service specification by Google.
- Sector: storage system in C++, claims to be 2x Hadoop.
- Eucalyptus: build-your-own-infrastructure open source software with compatible interface with Amazon EC2.
- etc.

Counting the numbers

Personal Computer

One : One

Client / Server

One : Many

Cloud Computing

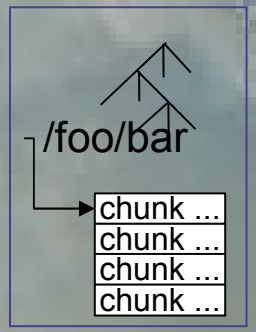
Many : Many

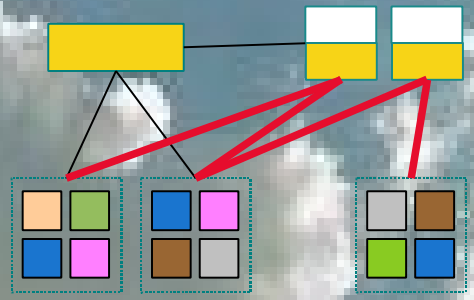
Developer transition

What Powers Google's Cloud Computing?

Commodity Hardware

Infrastructure Software





Performance: single machine not interesting
Reliability:

- Most reliable hardware will still fail: fault-tolerant software needed
- Fault-tolerant software enables use of commodity components

Standardization: use standardized machines to run all kinds of applications

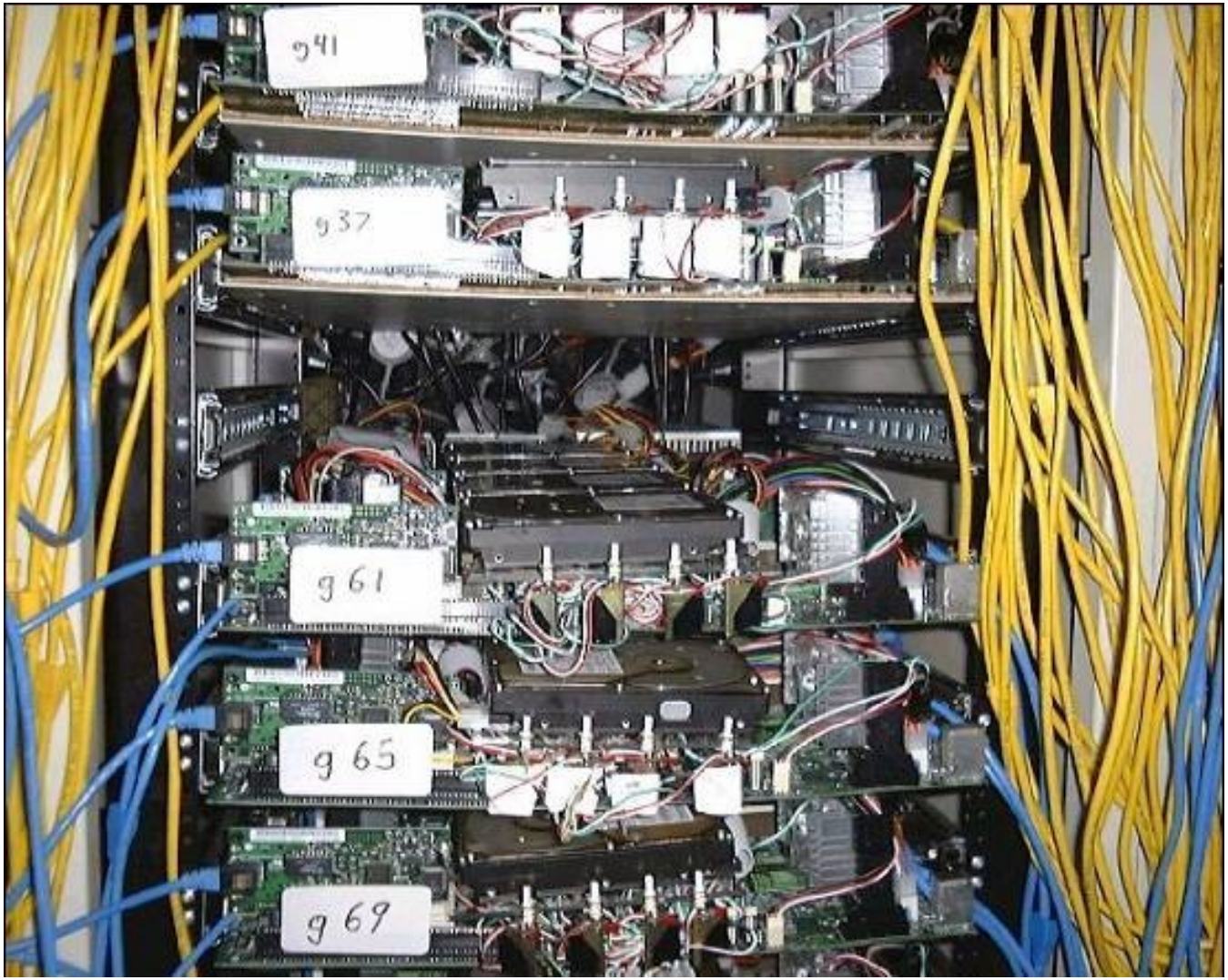
Distributed storage:
Google File System (GFS)

Distributed semi-structured data system: BigTable

Distributed data processing system: MapReduce

google.stanford.edu (circa 1997)

google.com (1999)



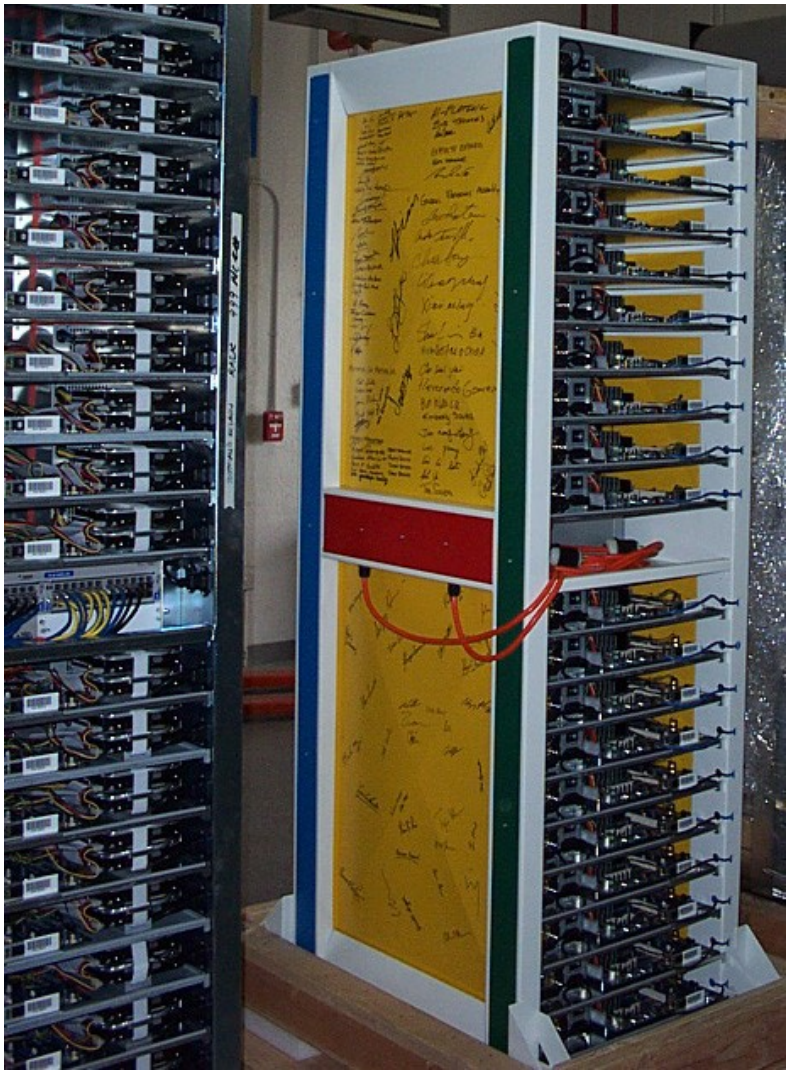
"cork boards"

Google Data Center (circa 2000)

google.com (new data center 2001)

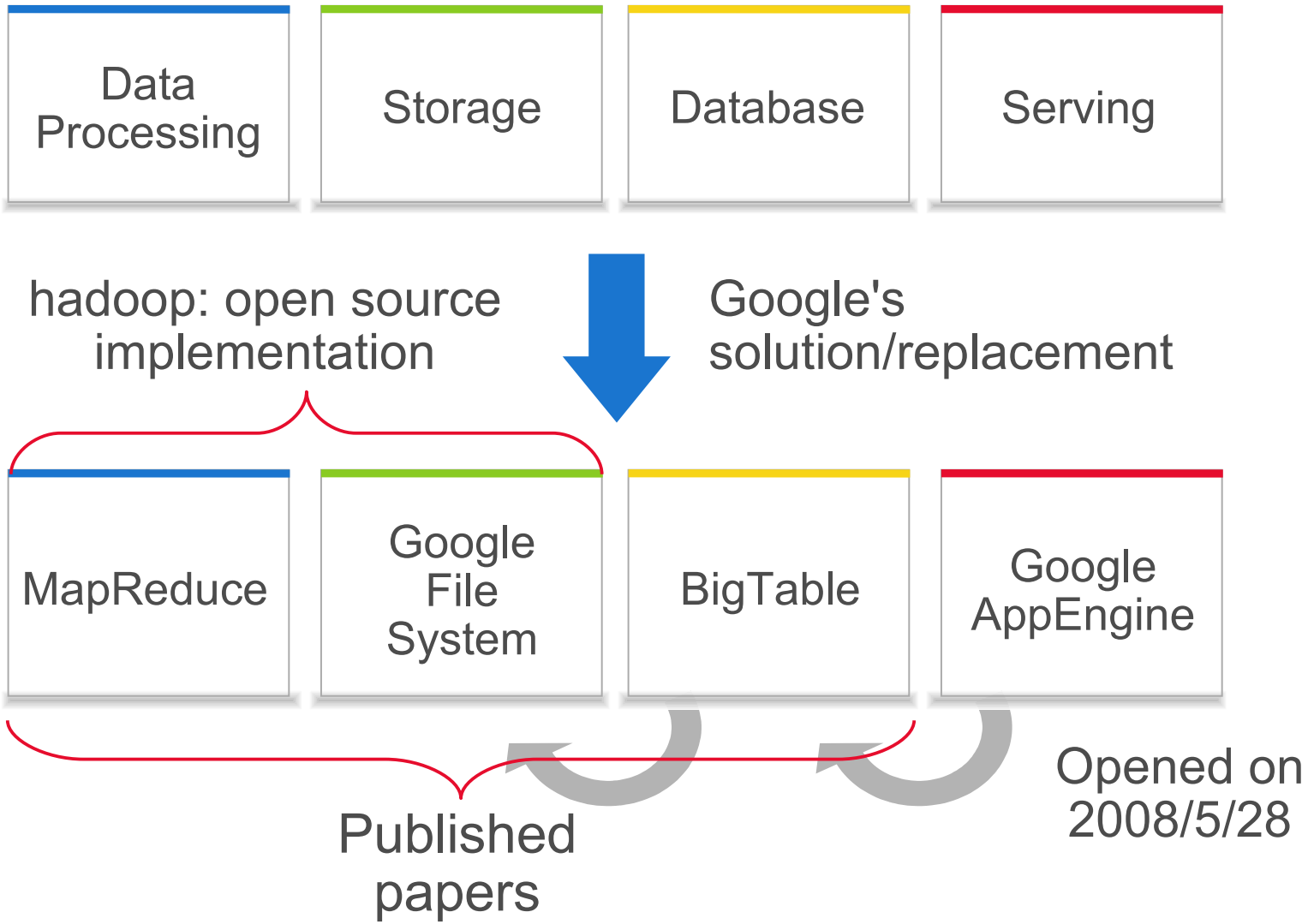
google.com (3 days later)

Current Design

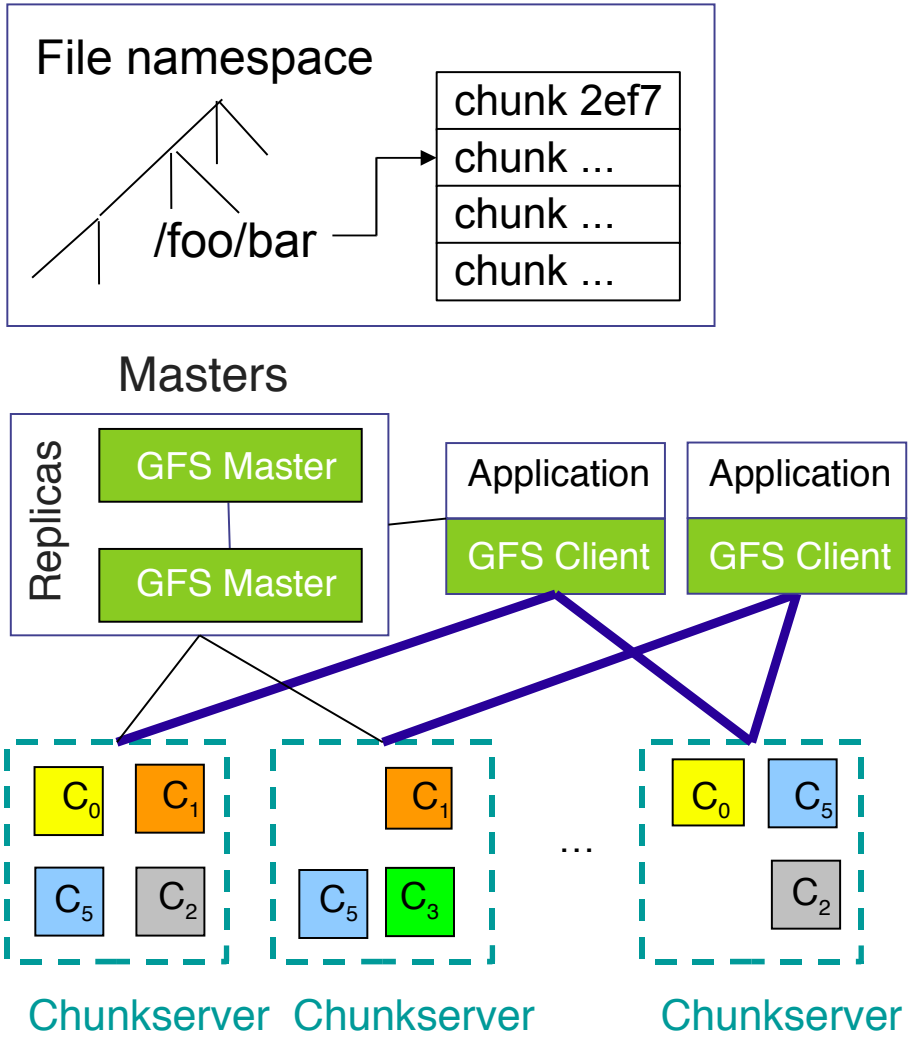


- In-house rack design
- PC-class motherboards
- Low-end storage and networking hardware
- Linux
- + in-house software

How to develop a web application that scales?



Google File System



- Files broken into chunks (typically 64 MB)
- Chunks triplicated across three machines for safety (tunable)
- Master manages metadata
- Data transfers happen directly between clients and chunkservers

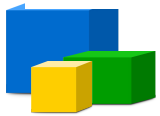
GFS Usage @ Google

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

- 200+ clusters
- Filesystem clusters of up to 5000+ machines
- Pools of 10000+ clients
- 5+ PB Filesystems

All in the presence of frequent HW failures

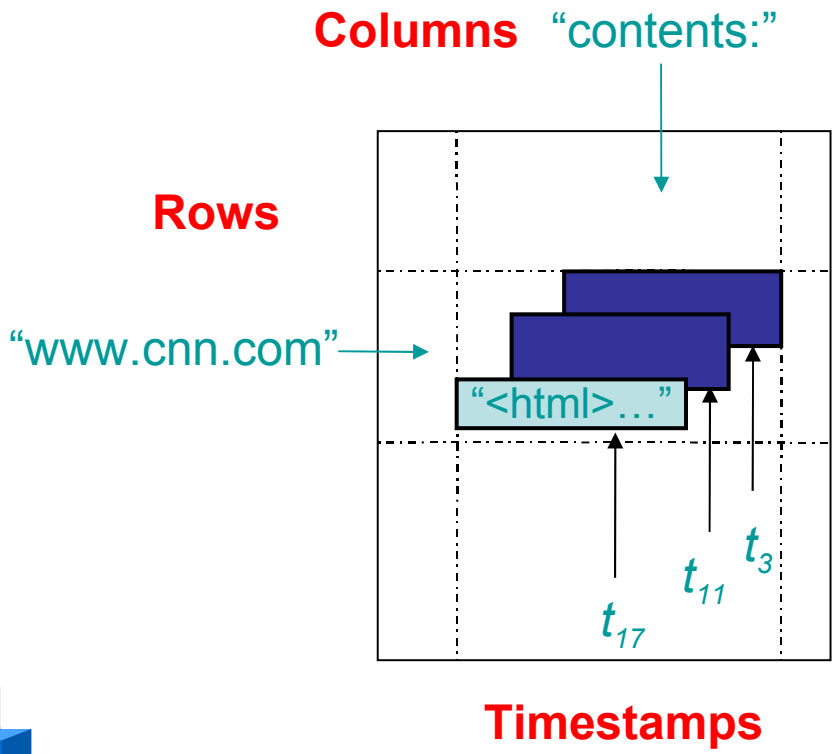


BigTable

Data model:

(row, column, timestamp)

→ cell contents

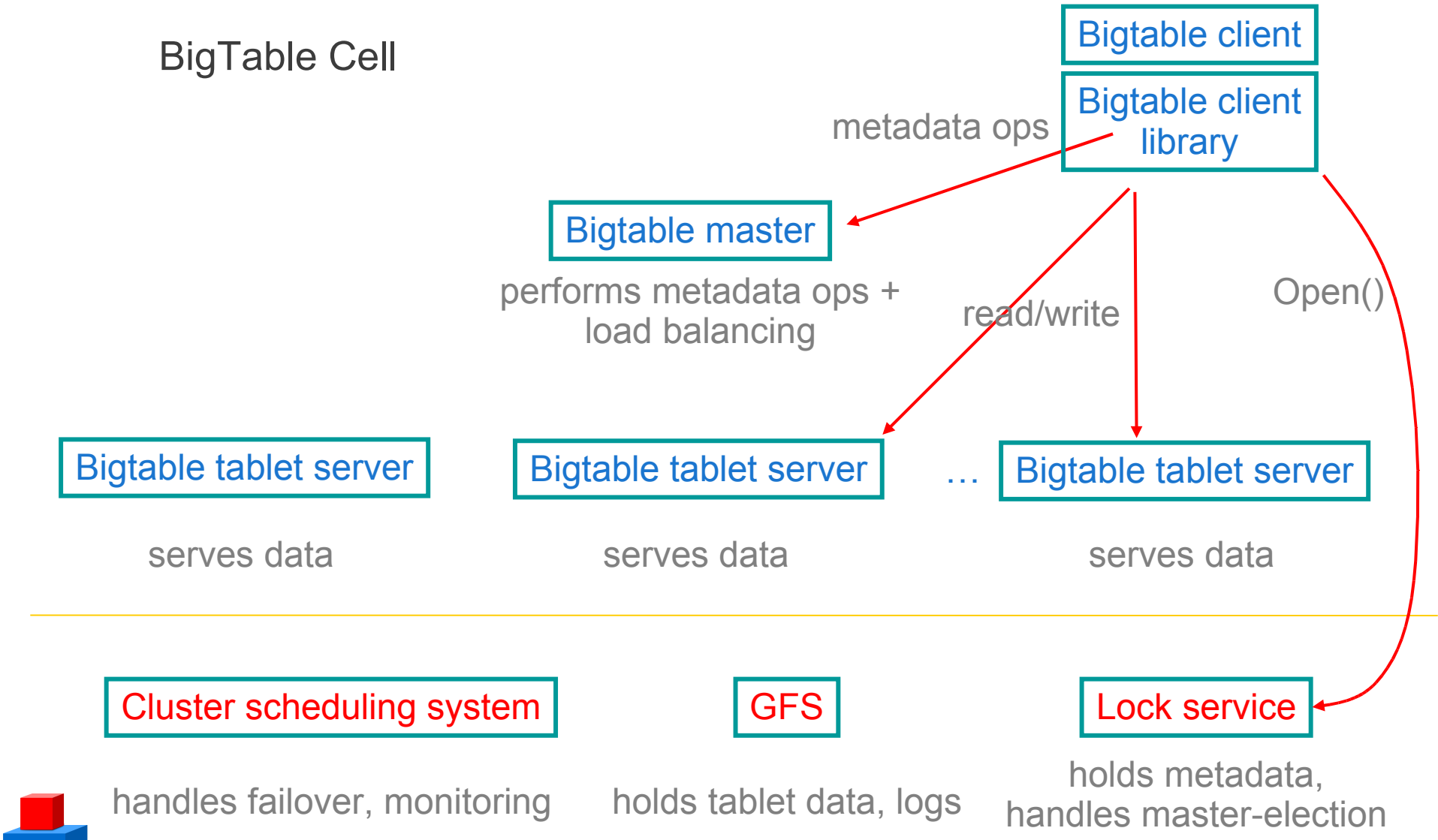


- Distributed multi-level sparse map: fault-tolerant, persistent
- Scalable:
  - Thousands of servers
  - Terabytes of in-memory data, petabytes of disk-based data
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance

Why not just use commercial DB?

- Scale is too large or cost is too high for most commercial databases
- Low-level storage optimizations help performance significantly
  - Much harder to do when running on top of a database layer
  - Also fun and challenging to build large-scale systems :)

System Structure



BigTable Summary

- Data model applicable to broad range of clients
  - Actively deployed in many of Google's services
- System provides high performance storage system on a large scale
  - Self-managing
  - Thousands of servers
  - Millions of ops/second
  - Multiple GB/s reading/writing
- Currently ~500 BigTable cells
- Largest bigtable cell manages ~3PB of data spread over several thousand machines (larger cells planned)

Distributed Data Processing

How do you process 1 month of apache logs to find the usage pattern numRequest[minuteOfTheWeek]?

- Input files: N rotated logs
- Size: O(TB) for popular sites – multiple physical disks
- Processing phase 1: launch M processes
  - input: N/M log files
  - output: one file of numRequest[minuteOfTheWeek]
- Processing phase 2: merge M output files of step 1

Pseudo Codes for Phase 1 and 2

```
def findBucket(requestTime):
  # return minute of the week

numRequest = zeros(1440*7)   # an array of 1440*7 zeros
for filename in sys.argv[2:]:
  for line in open(filename):
    minuteBucket = findBucket(findTime(line))
    numRequest[minuteBucket] += 1
outFile = open(sys.argv[1], 'w')
for i in range(1440*7):
  outFile.write("%d %d\n" % (i, numRequest[i]))
outFile.close()
```

```
numRequest = zeros(1440*7)   # an array of 1440*7 zeros
for filename in sys.argv[2:]:
  for line in open(filename):
    col = line.split()
    [i, count] = [int(col[0]), int(col[1])]
    numRequest[i] += count
# write out numRequest[] like phase 1
```

Task Management

- Logistics:
  - Decide which computers to run phase 1, make sure the log files are accessible (NFS-like or copy)
  - Similar for phase 2
- Execution:
  - Launch the phase 1 programs with appropriate command line flags, re-launch failed tasks until phase 1 is done
  - Similar for phase 2
- Automation: build task scripts on top of existing batch system (PBS, Condor, GridEngine, LoadLeveler, etc)

Technical Issues

- File management: where to store files?
  - Store all logs on the same file server ➔ Bottleneck!
  - Distributed file system: opportunity to run locally
- Granularity: how to decide N and M?
  - Performance ↗ when M ↗ until M == N if no I/O contention
  - Can M > N? Yes! Careful log splitting. Is it faster?
- Job allocation: assign which task to which node?
  - Prefer local job: knowledge of file system
- Fault-recovery: what if a node crashes?
  - Redundancy of data a must
  - Crash-detection and job re-allocation necessary

Performance

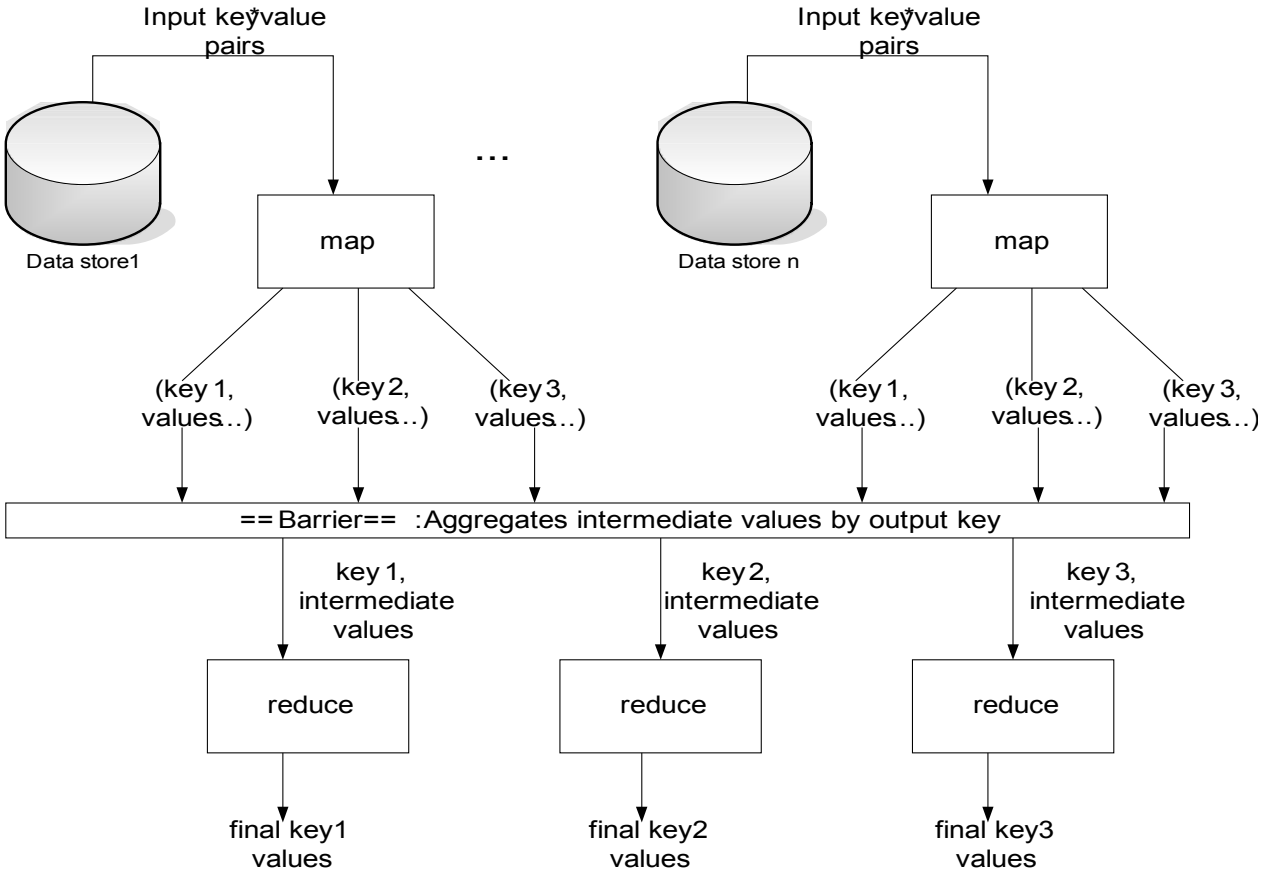
Robustness

Reusability

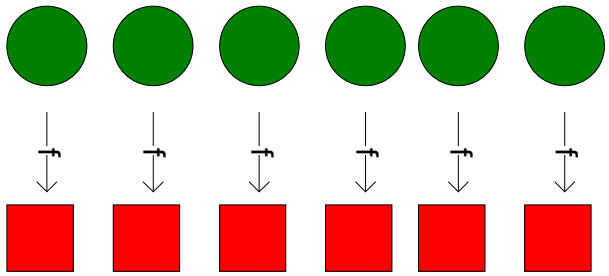
MapReduce – A New Model and System

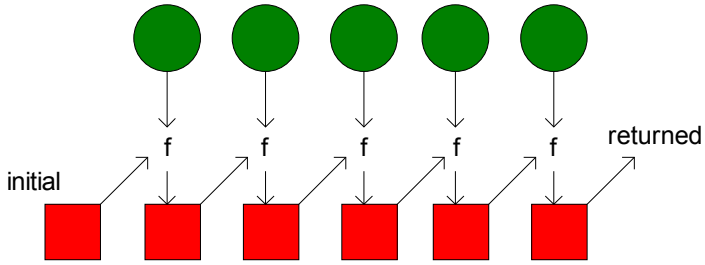
Two phases of data processing

- Map: (in_key, in_value) ➔ { (key_j, $value_j$) | j = 1, ..., K }
- Reduce: (key, [$value_1$, ... $value_L$]) ➔ (key, f_value)



MapReduce Programming Model





Borrowed from functional programming

map(f, [x1, x2, ...]) = [f(x1), f(x2), ...]

reduce(f, x0, [x1, x2, x3,...])
= reduce(f, f(x0, x1), [x2,...])
= ...
(continue until the list is exausted)

Users implement two functions:

map (in_key, in_value) ➔
(key_j, $value_j$) list

reduce [$value_1$, ... $value_L$] ➔ *f_value*

MapReduce Version of Pseudo Code

```
def findBucket(requestTime):
  # return minute of the week

class LogMinuteCounter(MapReduction):
  def Map(key, value, output):  # key is location
    minuteBucket = findBucket(findTime(value))
    output.collect(str(minuteBucket), "1")
  def Reduce(key, iter, output):
    sum = 0
    while not iter.done():
      sum += 1
    output.collect(key, str(sum))
```

- See, no file I/O!
- Only data processing logic...

  ... and gets much more than that!

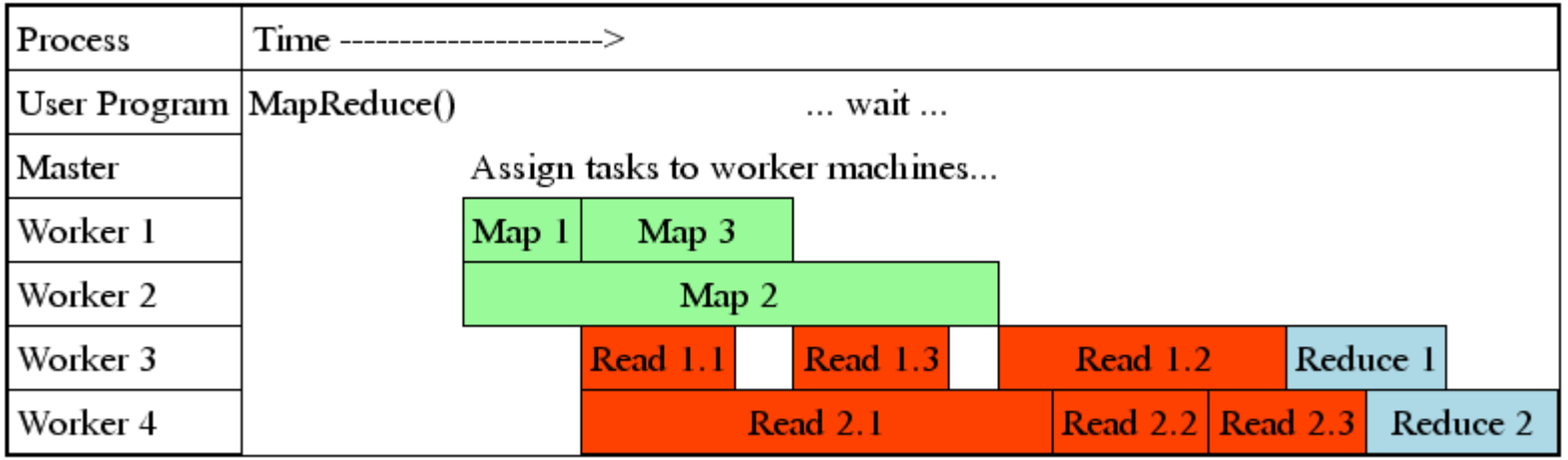
MapReduce Framework

For certain classes of problems, the MapReduce framework provides:

- Automatic & efficient parallelization/distribution
- I/O scheduling: Run mapper close to input data (same node or same rack when possible, with GFS)
- Fault-tolerance: restart failed mapper or reducer tasks on the same or different nodes
- Robustness: tolerate even massive failures, e.g. large-scale network maintenance: once lost 1800 out of 2000 machines
- Status/monitoring

Task Granularity And Pipelining

- Fine granularity tasks: many more map tasks than machines
  - Minimizes time for fault recovery
  - Can pipeline shuffling with map execution
  - Better dynamic load balancing
- Often use 200,000 map/5000 reduce tasks w/ 2000 machines

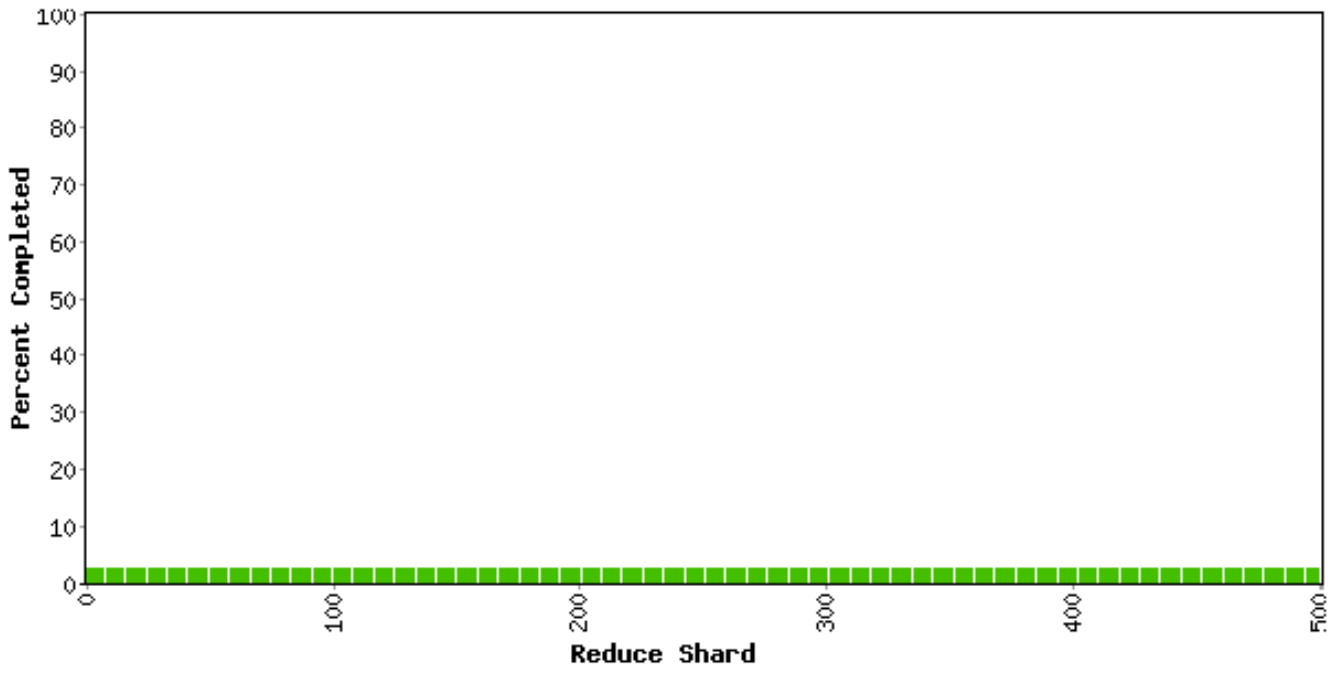


MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec

323 workers; 0 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	0	323	878934.6	1314.4	717.0
Shuffle	500	0	323	717.0	0.0	0.0
Reduce	500	0	0	0.0	0.0	0.0



Counters

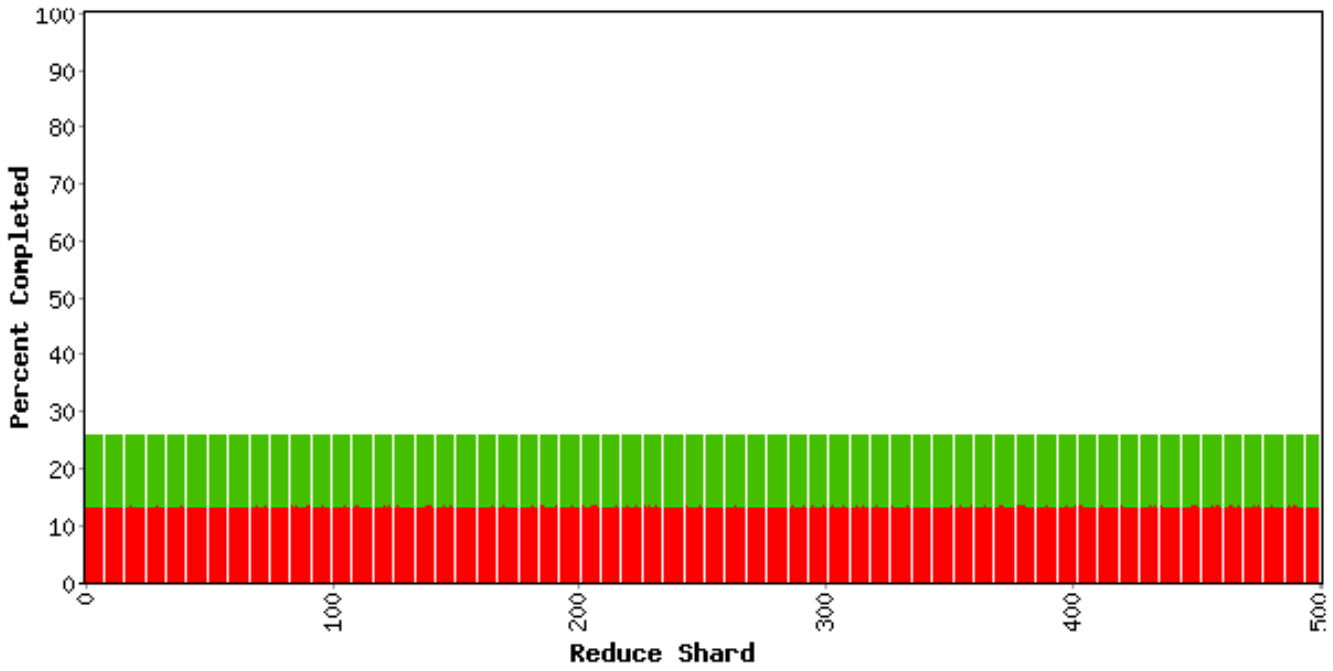
Variable	Minute
Mapped (MB/s)	72.5
Shuffle (MB/s)	0.0
Output (MB/s)	0.0
doc-index-hits	145825686
docs-indexed	506631
dups-in-index-merge	0
mr-operator-calls	508192
mr-operator-	506631

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	1857	1707	878934.6	191995.8	113936.6
Shuffle	500	0	500	113936.6	57113.7	57113.7
Reduce	500	0	0	57113.7	0.0	0.0



Counters

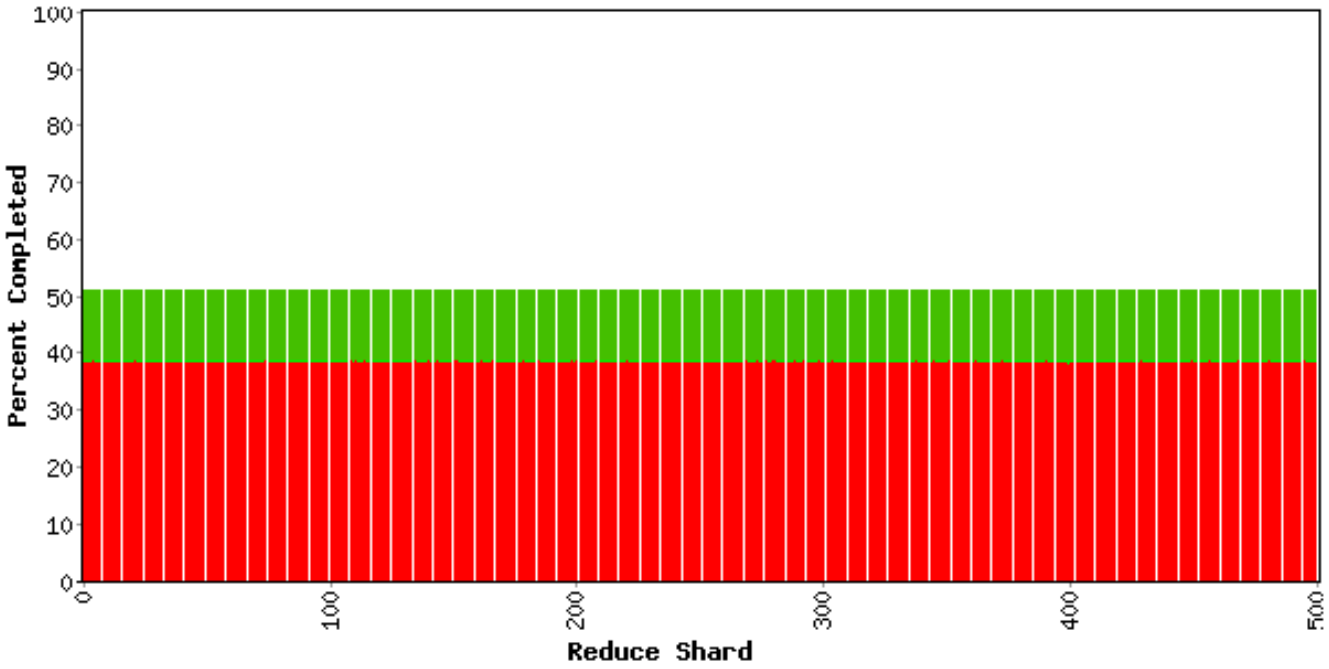
Variable	Minute
Mapped (MB/s)	699.1
Shuffle (MB/s)	349.5
Output (MB/s)	0.0
doc-index-hits	5004411944
docs-indexed	17290135
dups-in-index-merge	0
mr-operator-calls	17331371
mr-operator-outputs	17290135

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	5354	1707	878934.6	406020.1	241058.2
Shuffle	500	0	500	241058.2	196362.5	196362.5
Reduce	500	0	0	196362.5	0.0	0.0



Counters

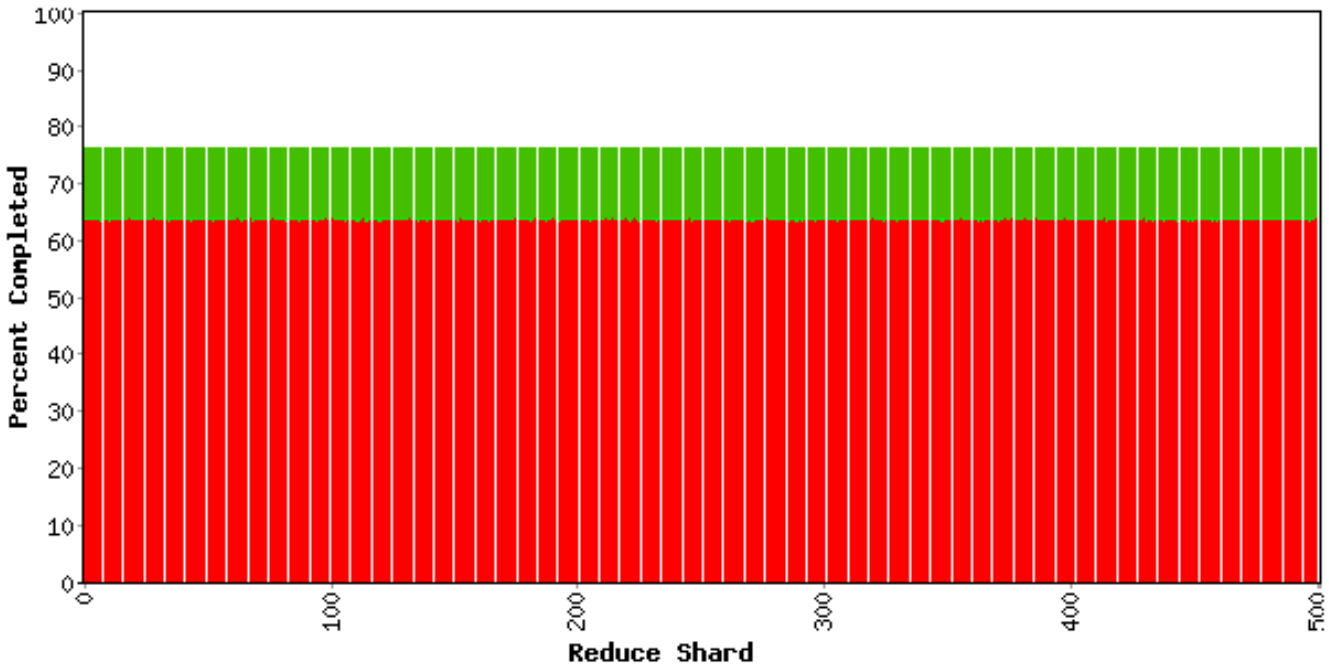
Variable	Minute
Mapped (MB/s)	704.4
Shuffle (MB/s)	371.9
Output (MB/s)	0.0
doc-index-hits	5000364228
docs-indexed	17300709
dups-in-index-merge	0
mr-operator-calls	17342493
mr-operator-outputs	17300709

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	8841	1707	878934.6	621608.5	369459.8
Shuffle	500	0	500	369459.8	326986.8	326986.8
Reduce	500	0	0	326986.8	0.0	0.0



Counters

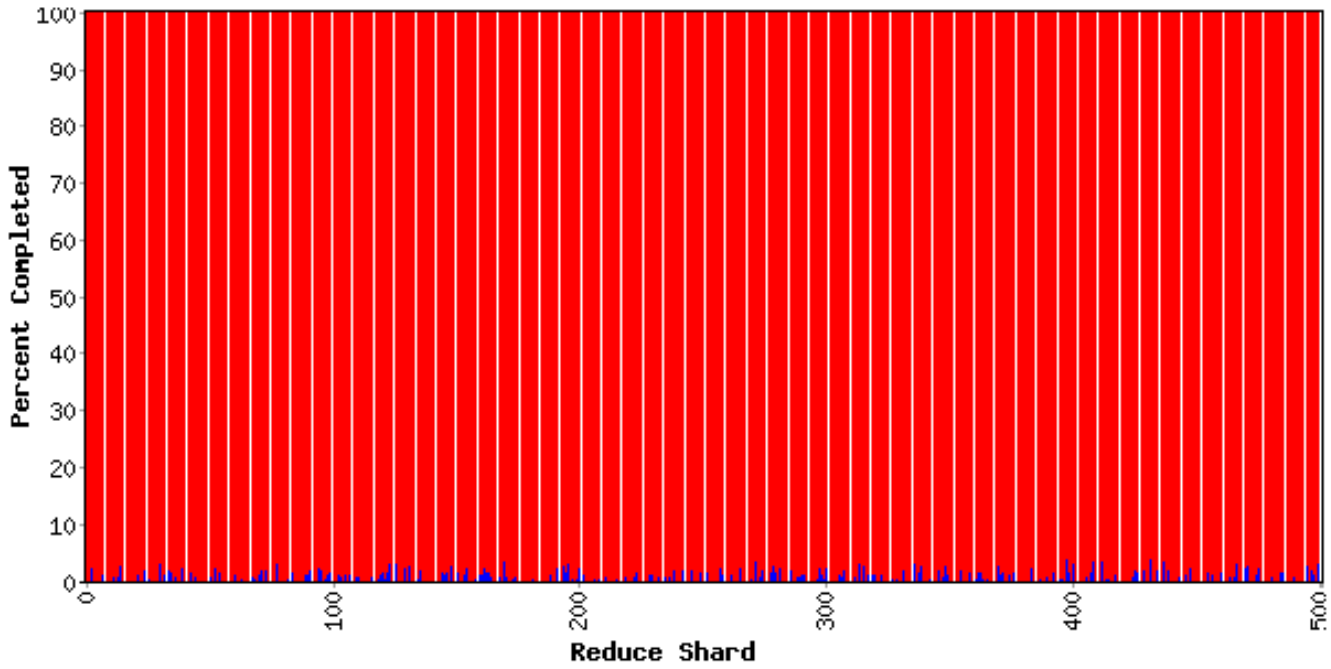
Variable	Minute
Mapped (MB/s)	706.5
Shuffle (MB/s)	419.2
Output (MB/s)	0.0
doc-index-hits	4982870667
docs-indexed	17229926
dups-in-index-merge	0
mr-operator-calls	17272056
mr-operator-outputs	17229926

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 29 min 45 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	195	305	523499.2	523389.6	523389.6
Reduce	500	0	195	523389.6	2685.2	2742.6



Counters

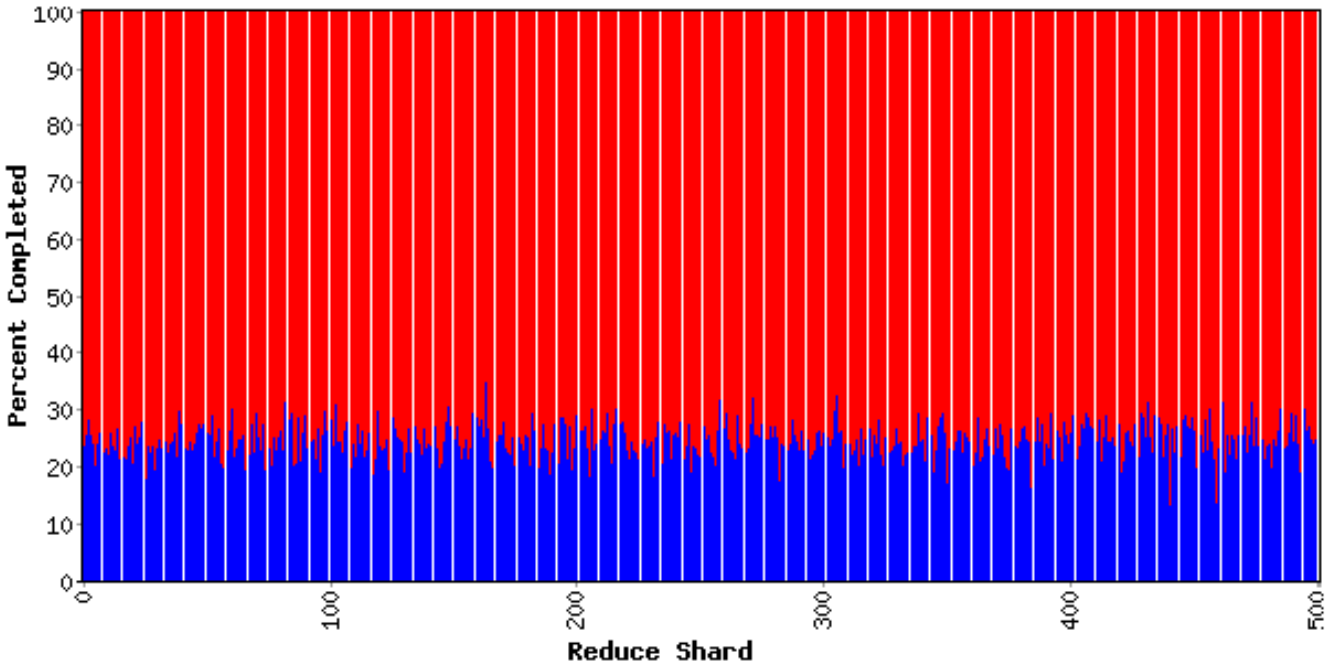
Variable	Minute	
Mapped (MB/s)	0.3	
Shuffle (MB/s)	0.5	
Output (MB/s)	45.7	
doc-index-hits	2313178	1056
docs-indexed	7936	3
dups-in-index-merge	0	
mr-merge-calls	1954105	
mr-merge-outputs	1954105	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	133837.8	136929.6



Counters

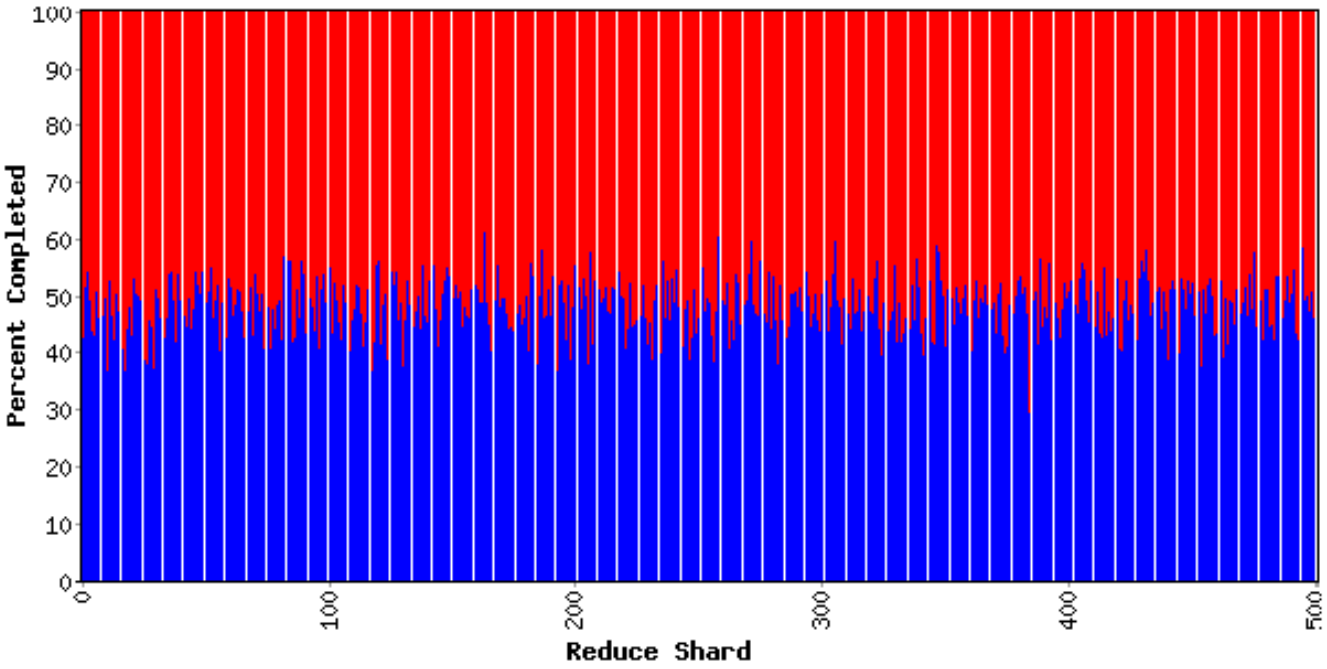
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.1	
Output (MB/s)	1238.8	
doc-index-hits	0	10
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	51738599	
mr-merge-outputs	51738599	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 33 min 22 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	263283.3	269351.2



Counters

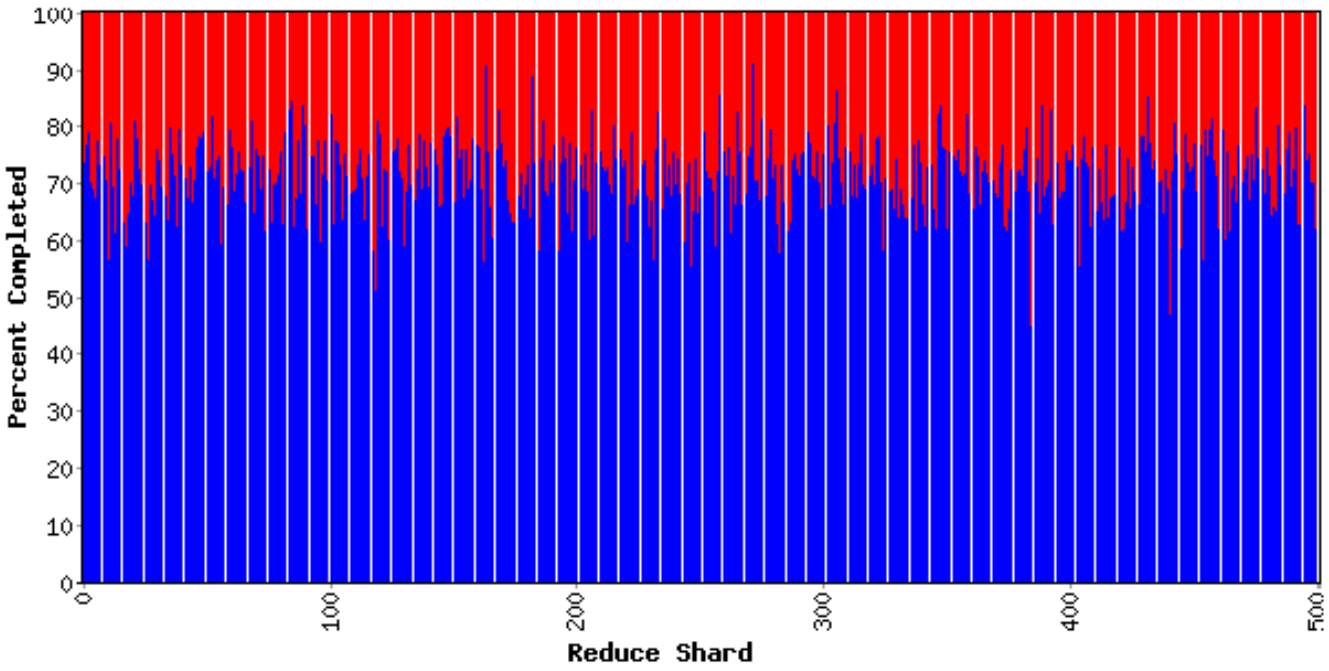
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	1225.1	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	51842100	
mr-merge-outputs	51842100	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 35 min 08 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	390447.6	399457.2



Counters

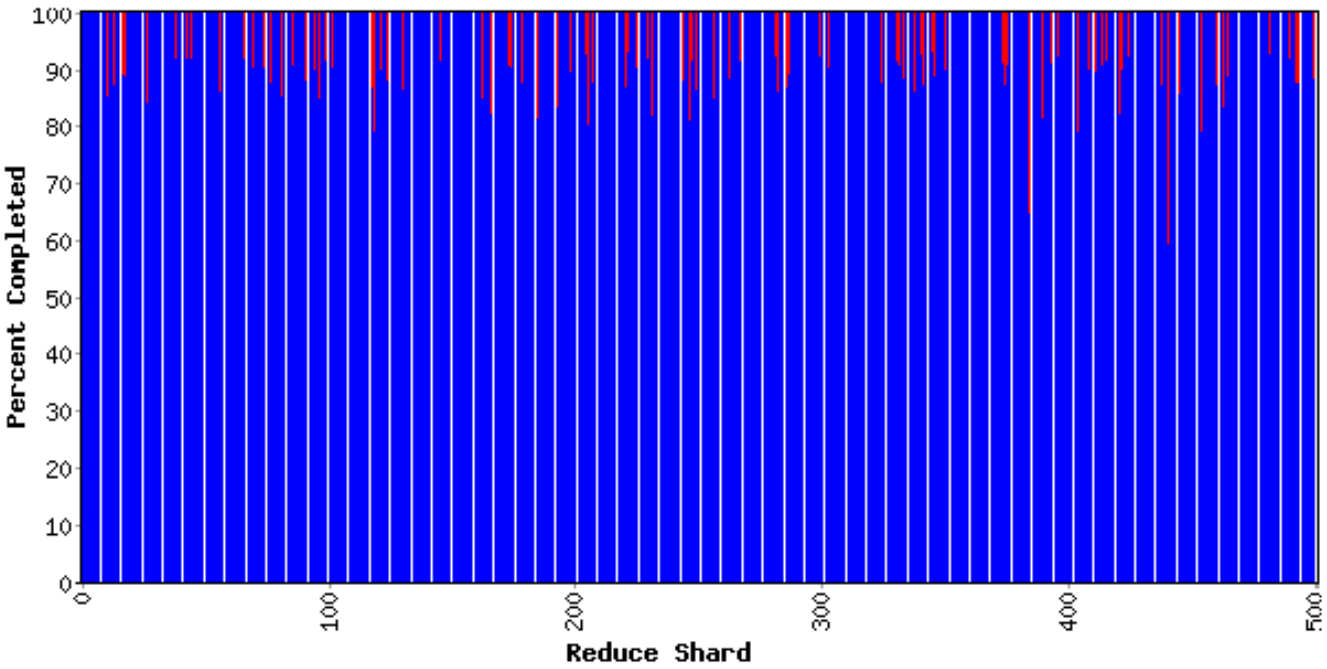
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	1222.0	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	51640600	
mr-merge-outputs	51640600	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	520468.6	520468.6
Reduce	500	406	94	520468.6	512265.2	514373.3



Counters

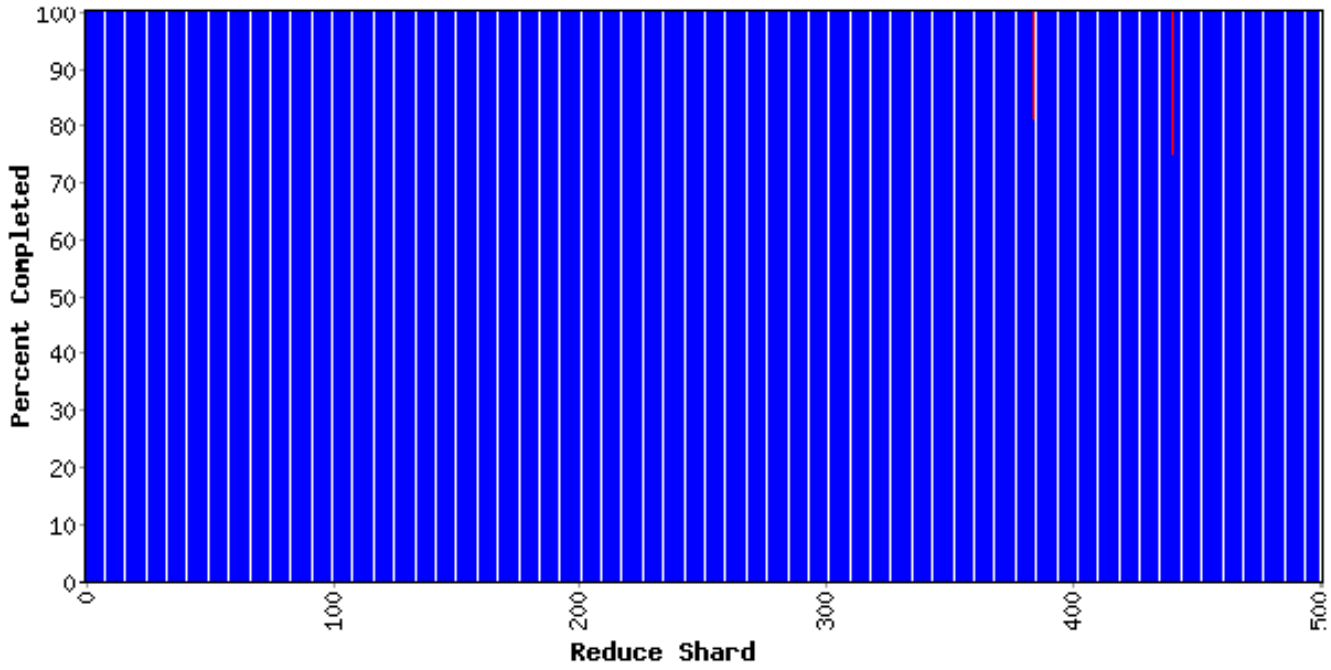
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	849.5	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	35083350	
mr-merge-outputs	35083350	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519781.8	519781.8
Reduce	500	498	2	519781.8	519394.7	519440.7



Counters

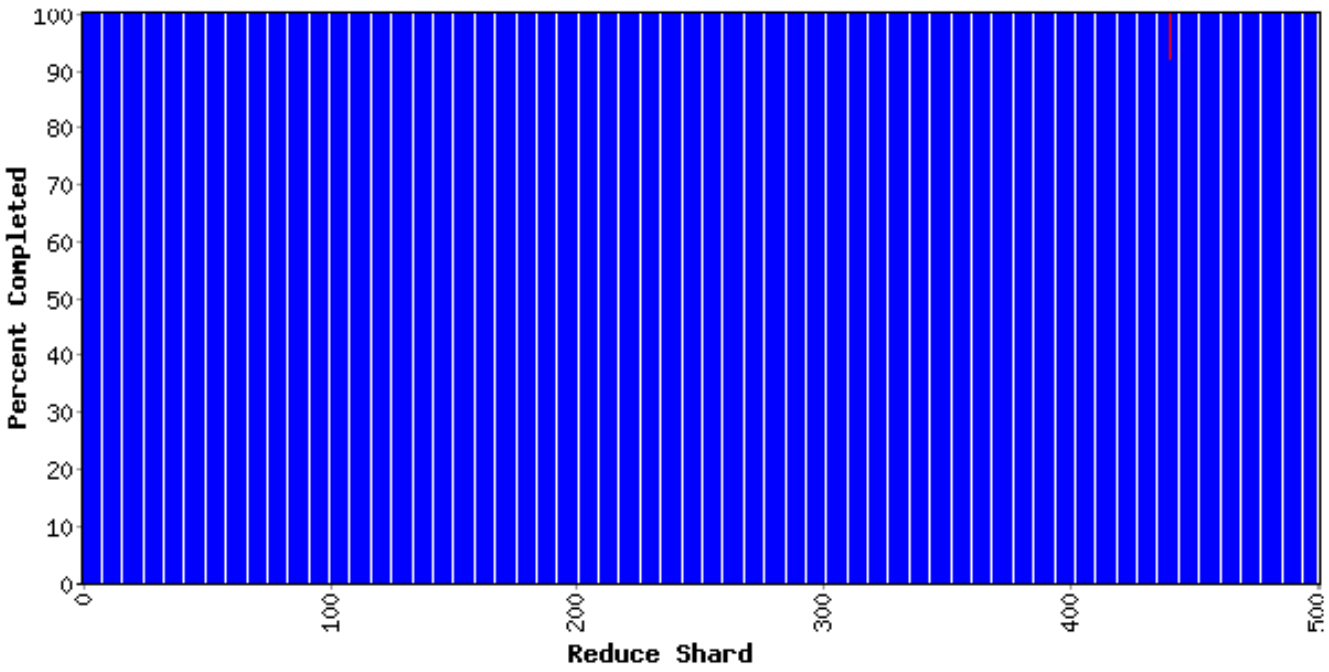
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	9.4	
doc-index-hits	0	10560
docs-indexed	0	36
dups-in-index-merge	0	
mr-merge-calls	394792	36
mr-merge-outputs	394792	36

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 40 min 43 sec

1707 workers; 1 deaths

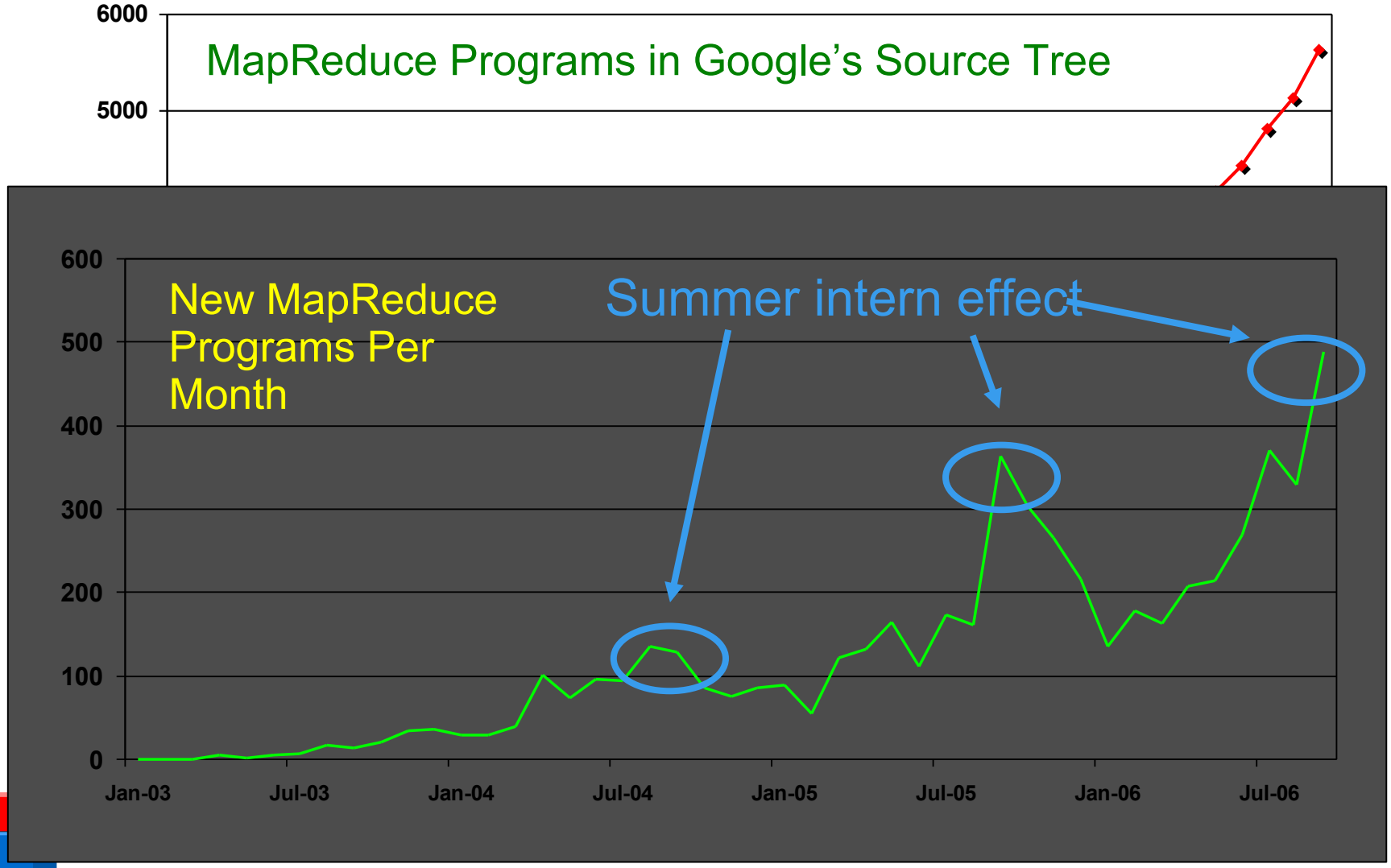
Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519774.3	519774.3
Reduce	500	499	1	519774.3	519735.2	519764.0



Counters

Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	1.9	
doc-index-hits	0	10560
docs-indexed	0	3
dups-in-index-merge	0	
mr-merge-calls	73442	3
mr-merge-outputs	73442	3

MapReduce: Adoption at Google



MapReduce: Uses at Google

- Typical configuration: 200,000 mappers, 500 reducers on 2,000 nodes
- Broad applicability has been a pleasant surprise
  - Quality experiments, log analysis, machine translation, ad-hoc data processing, …
  - Production indexing system: rewritten w/ MapReduce
    - ~10 MapReductions, much simpler than old code

	2004/08	2005/03	2006/03
Number of jobs	29,423	72,229	171,834
Avg completion time (sec)	634	934	874
Machine years used	217	981	2,002
Input data read (TB)	3,288	12,571	52,254
Intermediate data (TB)	758	2,756	6,743
Output data written (TB)	193	941	2,970
Avg workers per job	157	232	268
Avg worker deaths per job	1.2	1.9	5

MapReduce Summary

- MapReduce has proven to be a useful abstraction
- Greatly simplifies large-scale computations at Google
- Fun to use: focus on problem, let library deal with messy details

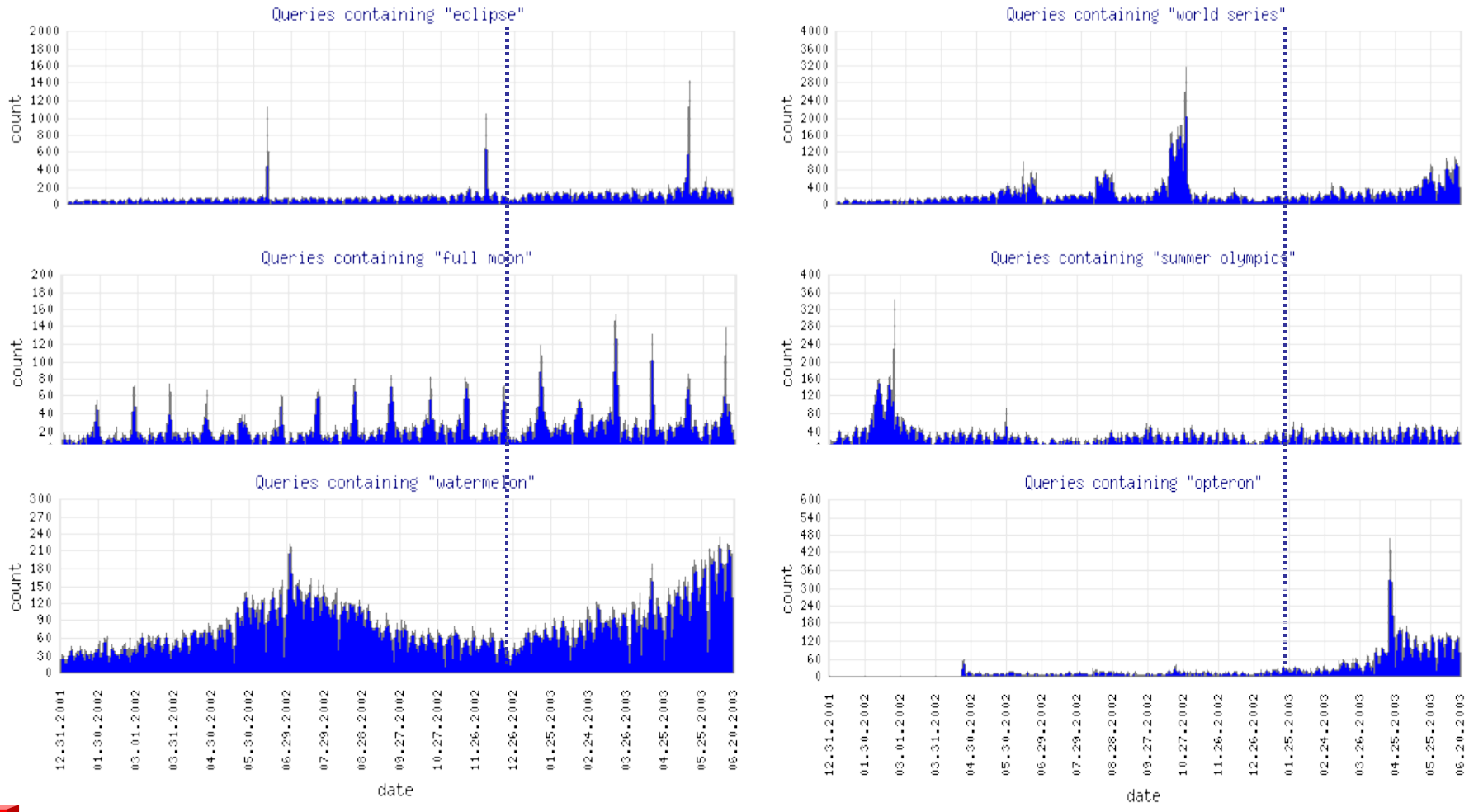
- Published

A Data Playground

MapReduce + BigTable + GFS = Data playground

- Substantial fraction of internet available for processing
- Easy-to-use teraflops/petabytes, quick turn-around
- Cool problems, great colleagues

Query Frequency Over Time



Learning From Data

Searching for Britney Spears...

britany spears
britny spears
briteny spears
britteny spears
briney spears
brittny spears
brintey spears
britanny spears
britiny spears
britnet spears
britiney spears
britaney spears
britnay spears
brithney spears
brtiney spears
britaney spears
britnay spears
brithney spears
brtiney spears
birtney spears
brintney spears
briteney spears
bitney spears
brinty spears
brittaney spears
brittnay spears
britey spears
brittiny spears
brtney spears
bretney spears
britneys spears
britne spears

488941 britney spears
40134 brittany spea
36315 brittney spea
24342 britany spear
7331 britny spears
6633 briteny spear
2696 britteny spea
1807 briney spears
1635 brittny spear
1479 brintey spear
1479 britanny spea
1338 britiny spear
1211 britnet spear
1095 britiney spea
991 britaney spea
991 britnay spear
811 brithney spea
811 brtiney spear
664 birtney spear
664 brintney spea
664 briteney spea
601 bitney spears
601 brinty spears
544 brittaney spe
544 brittnay spea
364 britey spears
364 brittiny spea
329 brtney spears
269 bretney spear
269 britneys spea
244 britne spears
244 brytney spear
220 breatney spea
220 britiany spea
199 britnney spea
163 britnry spear
147 breatny spear
147 brittiney spe
147 britty spears
147 brotney spear
147 brutney spear
133 britteney spe
133 briyney spear
121 bittany spear
121 bridney spears
121 britainy spears
121 britmey spears
109 brietney spears
109 brithny spears

17 brittanie spears
15 brinney spears
15 briten spears
15 briterney spears
15 britheny spears

9 brinttany spears
9 britanay spears
9 britinany spears
9 britn spears
9 britnew spears
9 britneyn spears
9 britrney spears
9 brtiny spears
9 brtittney spears
9 brtny spears
9 brytny spears
9 rbitney spears
8 birtiny spears
8 bithney spears
8 brattany spears
8 breitny spears
8 breteny spears
8 brightny spears
8 brintay spears
8 brinttey spears
8 briotney spears
8 britanys spears
8 britley spears
8 britneyb spears
8 britnrey spears
8 britnty spears
8 brittner spears
8 brottany spears
7 baritney spears
7 birntey spears
7 biteney spears
7 bitiny spears
7 breateny spears
7 brianty spears
7 brintye spears
7 britianny spears
7 britly spears
7 britnej spears
7 britneyu spears
7 britniey spears
7 britnnay spears
7 brittian spears
7 briyny spears
7 brrittany spears
7 brttiney spears
7 btiteny spears
7 btrittany spears
6 beritny spears
6 bhritney spears

5 brney spears
5 broitney spears
5 brotny spears
5 bruteny spears
5 btiyney spears
5 btrittney spears
5 gritney spears
5 spritney spears
4 bittny spears
4 bnritney spears
4 brandy spears
4 brbritney spears
4 breatiny spears
4 breetney spears
4 bretiney spears
4 brfitney spears
4 briattany spears
4 brieteny spears
4 briety spears
4 briitny spears
4 briittany spears
4 brinie spears
4 brinteney spears
4 brintne spears
4 britaby spears
4 britaey spears
4 britainey spears
4 britinie spears
4 britinney spears
4 britmney spears
4 britnear spears
4 britnel spears
4 britneuy spears
4 britnewy spears
4 britnmey spears
4 brittaby spears
4 brittery spears
4 britthey spears
4 brittnaey spears
4 brittnat spears
4 brittneny spears
4 brittnye spears
4 brittteny spears
4 briutney spears
4 briyeny spears
4 brnity spears
4 brtteny spears
4 brttiany spears
4 bryney spears

3 britiy spears
3 britmeny spears
3 britneeey spears
3 britnehy spears
3 britnely spears
3 britnesy spears
3 britnetty spears
3 britnex spears
3 britneyxxx spears
3 britnity spears
3 britntey spears
3 britnyey spears
3 britterny spears
3 brittneey spears
3 brittnney spears
3 brittnyey spears
3 brityen spears
3 briytney spears
3 brltney spears
3 broteny spears
3 brtaney spears
3 brtiiany spears
3 brtinay spears
3 brtinney spears
3 brtitany spears
3 brtiteny spears
3 brtnet spears
3 brytiny spears
3 btney spears
3 drittney spears
3 pretney spears
3 rbritney spears
2 barittany spears
2 bbbritney spears
2 bbitney spears
2 bbritny spears
2 bbrittany spears
2 beitany spears
2 beitny spears
2 bertney spears
2 bertny spears
2 betney spears
2 betny spears
2 bhriney spears
2 biney spears
2 bintey spears
2 biretny spears
2 biritany spears
2 birittany spears

2 brirreny spears
2 brirtany spears
2 brirttany spears
2 brirttney spears
2 britain spears
2 britane spears
2 britaneny spears
2 britania spears
2 britann spears
2 britanna spears
2 britannie spears
2 britannt spears
2 britannu spears
2 britanyl spears
2 britanyt spears
2 briteeny spears
2 britenany spears
2 britenet spears
2 briteniy spears
2 britenys spears
2 britianey spears
2 britin spears
2 britinary spears
2 britmy spears
2 britnaney spears
2 britnat spears
2 britnbey spears
2 britndy spears
2 britneh spears
2 britneney spears
2 britneyб spears
2 britneye spears
2 britneyh spears
2 britneym spears
2 britneyyy spears
2 britnhey spears
2 britnjey spears
2 britnne spears
2 britnu spears
2 britoney spears
2 britrany spears
2 britreny spears
2 britry spears
2 britsany spears
2 brittanay spears
2 brittang spears
2 brittans spears
2 brittanyh spears
2 brittanyn spears

Google 其他的 Cloud Offerings

- 給開發者： App Engine

Lets you run your *web* application on top of Google's scalable infrastructure.

- 和大學合作： MapReduce

讓學生提早掌握大規模資料處理的技術， 包括

Google File System, MapReduce, BigTable

Are data centers environment-unfriendly?

http://www.youtube.com/watch?v=zRwPSFpLX8I

Clean Energy Initiatives

Power Usage Effectiveness (PUE) =

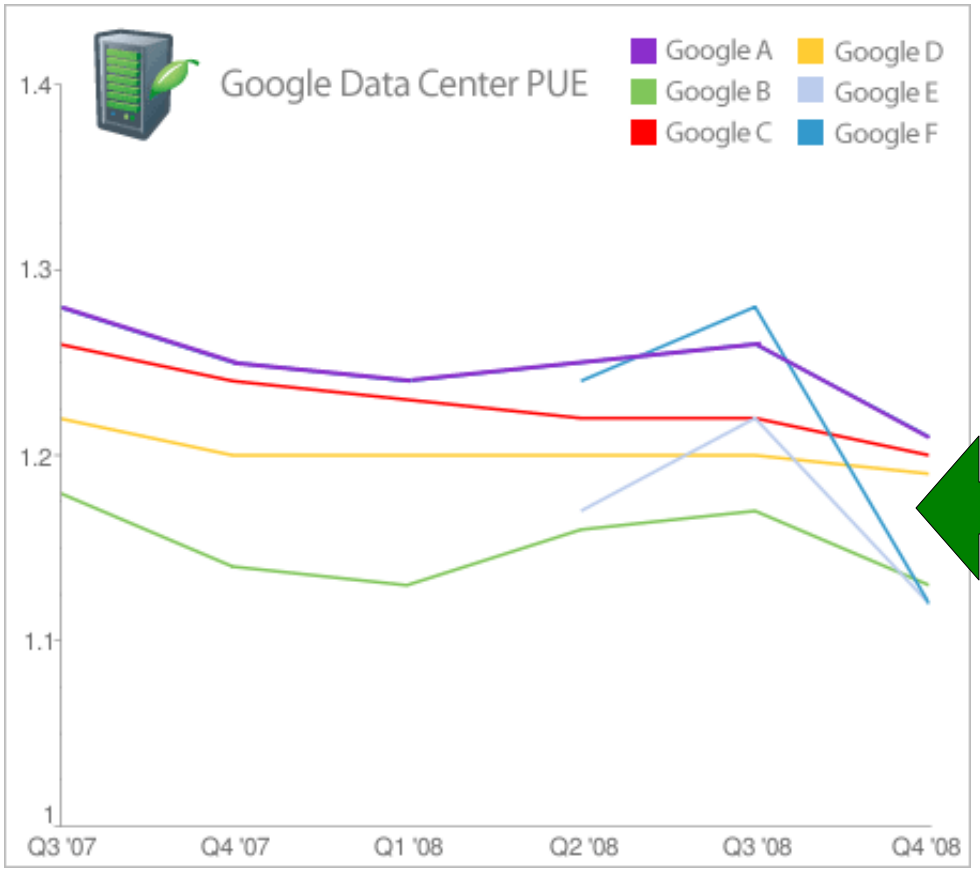
Total power consumption /

power consumed by IT equipments

In 2007, the EPA of the U.S. sent a report to the congress about the estimated PUE of data centers in year 2011:

Scenario	PUE
Current Trends	1.9
Improved Operations	1.7
Best Practices	1.3
State-of-the-Art	1.2

Google's data centers



6 Data centers with
> 5MW of power

mean PUE =
1.19

Details: http://www.google.com/corporate/green/datacenters/measuring.html

Summary

- More and more applications and data are moving to web servers
- More and more user rely on web access to find information, do their jobs, talk with their friends, publish their articles, pictures, video, etc.
- and they want to do so, not limited to his desk
- and that's Cloud Computing