Distributed Coordination via ZooKeeper

Flavio Junqueira
Yahoo! Research, Barcelona

Hadoop in China 2011

A bit of history

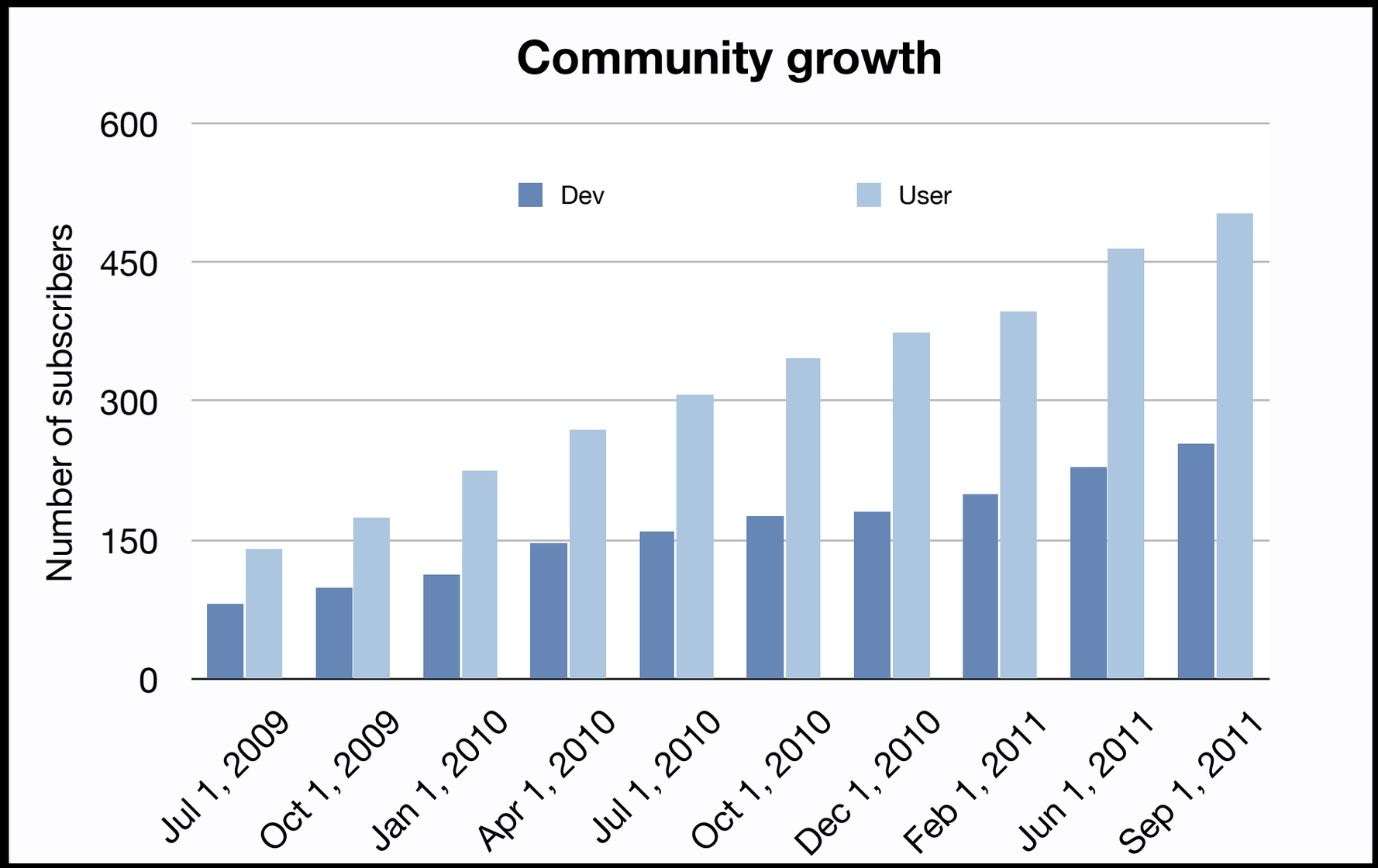
June 2007: Early adopters: Message Broker, Crawler

Oct 2007: Sourceforge

June 2008: Move to Apache, subproject of Hadoop

Nov 2010: Top level project

Apache community growth



Widely used

Widely used

{ 2010 08 26 }

Zookeeper experience

While working on Kafka, a distributed pub/sub system (more on that later) at LinkedIn, I need to use Zookeeper (ZK) to implement the load-balancing logic. I'd like to share my experience of using Zookeeper. First of all, for those of you who don't know, Zookeeper is an Apache project that implements a consensus service based on a variant of Paxos (it's similar to Google's Chubby). ZK has a very simple, file system like API. One can create a path, set the value of a path, read the value of a path, delete a path, and list the children of a path. ZK does a couple of more interesting things: (a) one can register a watcher on a path and get notified when the children of a path or the value of a path is changed, (b) a path can be created as ephemeral, which means that if the client that created the path is gone, the path is automatically removed by the ZK server. However, don't let the simple API fool you. One needs to understand a lot more than those APIs in order to use them properly. For me, this translates to weeks asking the ZK mailing list (which is pretty responsive) and our local ZK experts.

Jun Rao, LinkedIn

Widely used

{ 2010 08 26 }

Zookeeper experience

While working on Kafka, a distributed pub/sub system (more on that later) at LinkedIn, I need to use Zookeeper (ZK) to implement the load-balancing logic. I'd like to share my experience of using Zoo
on a variant of Paxos (it's similar t
path, delete a path, and list the ch
children of a path or the value of a
automatically removed by the ZK s
properly. For me, this translates to

Since Messages accepts data from many sources such as email and SMS, we decided to write an application server from scratch instead of using our generic Web infrastructure to handle all decision making for a user's messages. It interfaces with a large number of other services: we store attachments in Haystack, wrote a user discovery service on top of Apache ZooKeeper, and talk to other infrastructure services for email account verification, friend relationships, privacy decisions, and delivery decisions (for example, should a message be sent over chat or SMS). We spent a lot of time making sure each of these services are reliable, robust, and performant enough to handle a real-time messaging system.

Kannan Muthukkaruppan, Facebook

Widely used

{ 2010 08 26 }

Zookeeper experience

While working on Kafka, a distributed pub/sub system (more on that later) at LinkedIn, I need to use Zookeeper (ZK) to implement the load-balancing logic. I'd like to share my experience of using Zoo
on a variant of Paxos (it's similar t
path, delete a path, and list the ch
children of a path or the value of a
automatically removed by the ZK s
properly. For me, this translates to

Since Messages accepts data from many sources such as email and SMS, we decided to write an application server from scratch instead of using our generic Web infrastructure to handle all decision making for a user's messages. It interfaces with a large number of other services: we store attachments in Haystack, wrote a user discovery service on top of Apache

- ResourceManager - The ResourceManager uses Apache ZooKeeper for fail-over. When the ResourceManager fails, a secondary can quickly recover via cluster state saved in ZooKeeper. The ResourceManager, on a fail-over, restarts all of the queued and running applications.

Arun Murty, NextGen Hadoop

Widely used

{ 2010 08 26 }

Zookeeper experience

While working on Kafka, a distributed pub/sub system (more on that later) at LinkedIn, I need to use Zookeeper (ZK) to implement the load-balancing logic. I'd like to share my experience of using Zoo
on a variant of Paxos (it's similar t
path, delete a path, and list the ch
children of a path or the value of a
automatically removed by the ZK s
properly. For me, this translates to

Since Messages accepts data from many sources such as email and SMS, we decided to write an application server from scratch instead of using our generic Web infrastructure to handle all decision making for a user's messages. It interfaces with a large number of other services: we store attachments in Haystack, wrote a user discovery service on top of Apache

- ResourceManager - The ResourceManager uses Apache ZooKeeper for fail-over. When the ResourceManager fails, a secondary can quickly recover via cluster state saved in ZooKeeper
applications

The Apache Software Foundation has a neat tool for distributed lock services, called Zookeeper. Scalr based its distributed cron jobs off of it, so that users can setup scripts to be executed periodically, like cron jobs, without running the risk of multiple executions or failure to execute.

Sebastian Stadil, Scalr

Widely used

{ 2010 08 26 }

Zookeeper experience

While working on Kafka, a distributed pub/sub system (more on that later) at LinkedIn, I need to use Zookeeper (ZK) to implement the load-balancing logic. I'd like to share my experience of using Zoo
on a variant of Paxos (it's similar t
path, delete a path, and list the ch
children of a path or the value of a
automatically removed by the ZK s
properly. For me, this translates to

Since Messages accepts data from many sources such as email and SMS, we decided to write an application server from scratch instead of using our generic Web infrastructure to handle all decision making for a user's messages. It interfaces with a large number of other services: we store attachments in Haystack, wrote a user discovery service on top of Apache

- ResourceManager - The ResourceManager uses Apache ZooKeeper for fail-over. When the ResourceManager fails, a secondary can quickly recover via cluster state saved in ZooKeeper
applications

The Apache Software Foundation has a neat tool for distributed lock services, called Zookeeper. Scalr based its distributed cron jobs off of it, so that users can setup scripts to be executed periodically, like cron jobs, without running the risk of multiple executions or failure to execute.

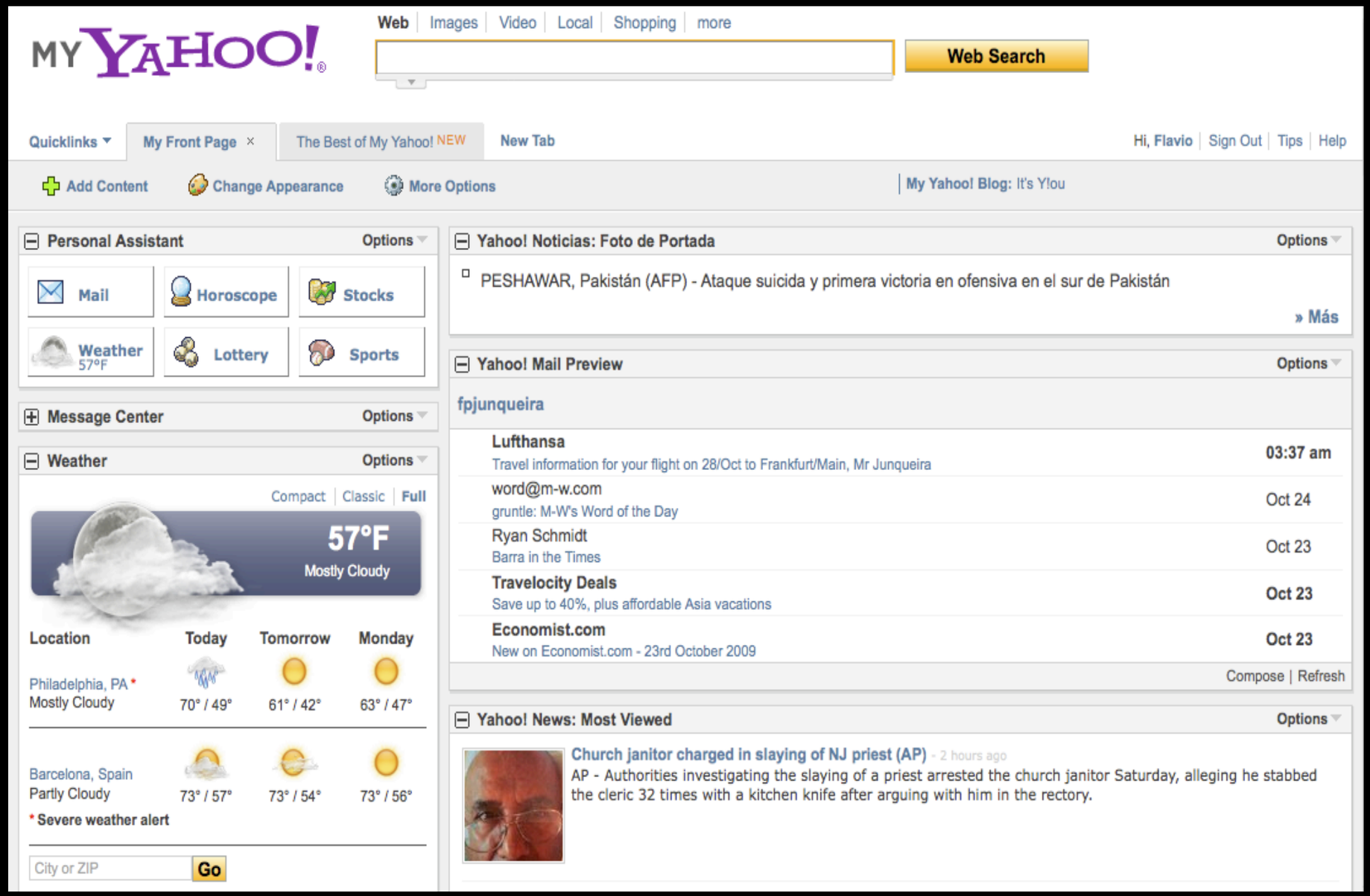
Operating System and Configuration

Digg runs on Debian stable based GNU/Linux servers which we configure with Clusto, Puppet and using a configuration system over Zookeeper.

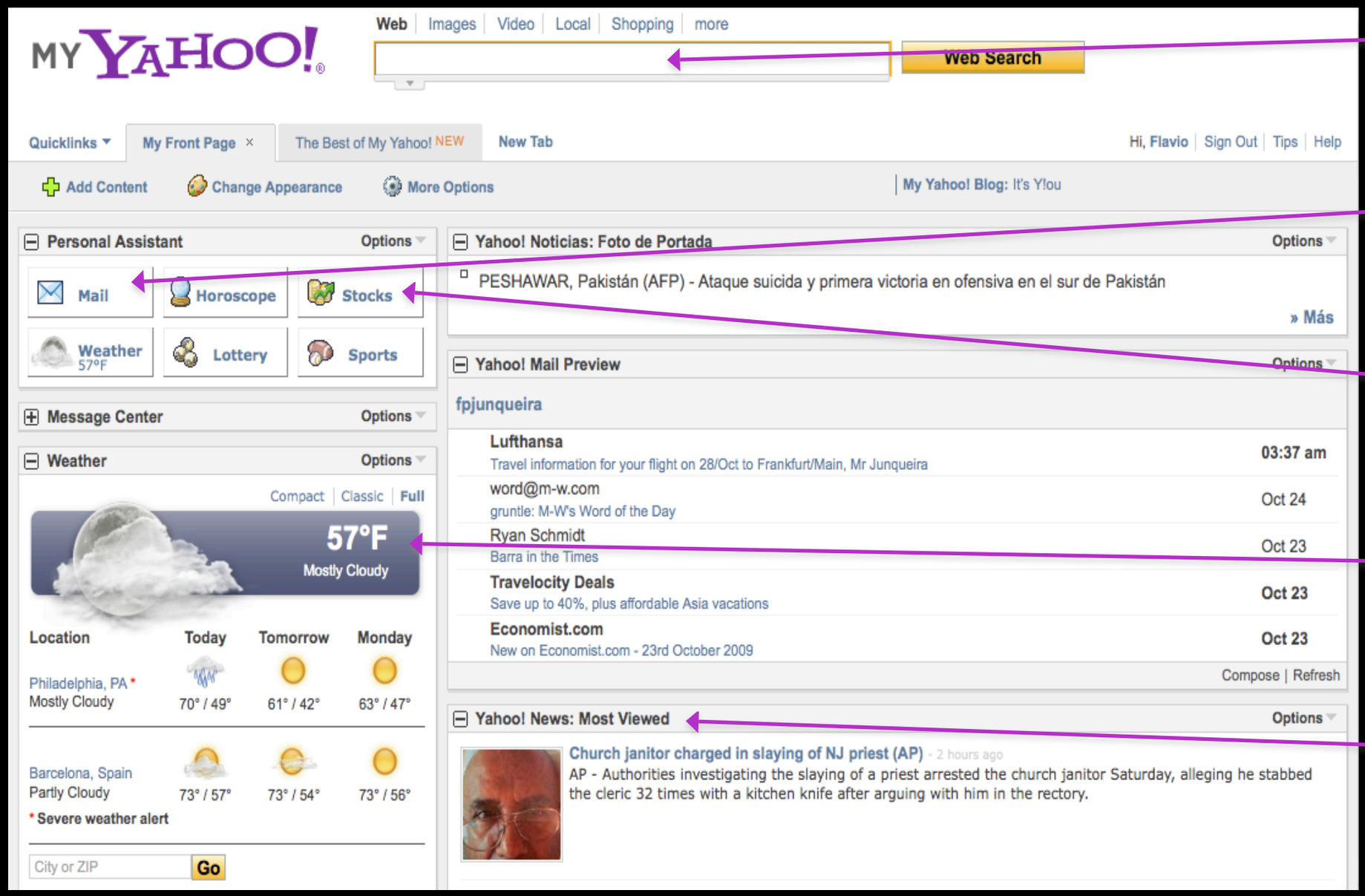
Dave Beckett, Digg.com

Where we are coming from...

Yahoo! Portal



Yahoo! Portal



Search

E-mail

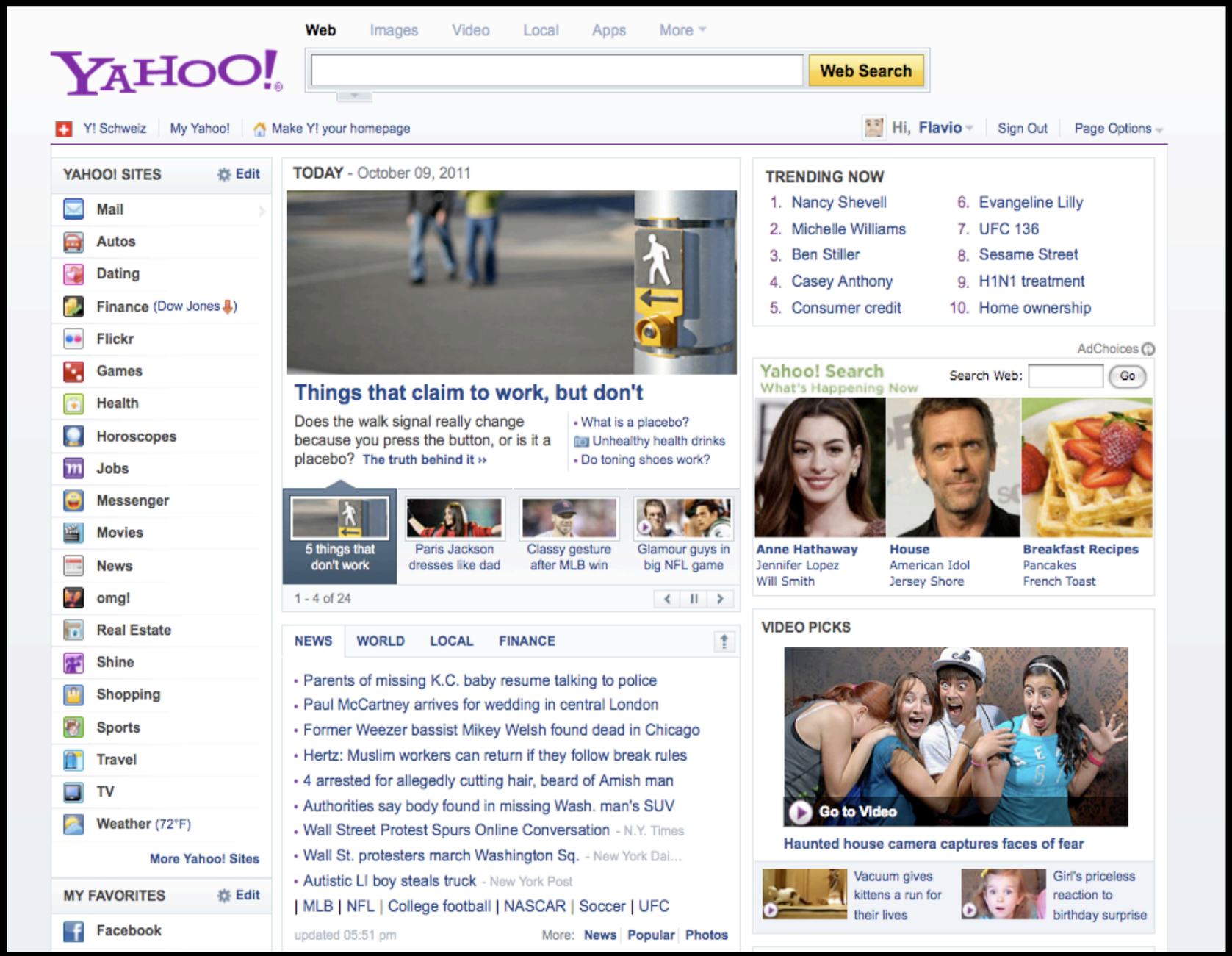
Finance

Weather

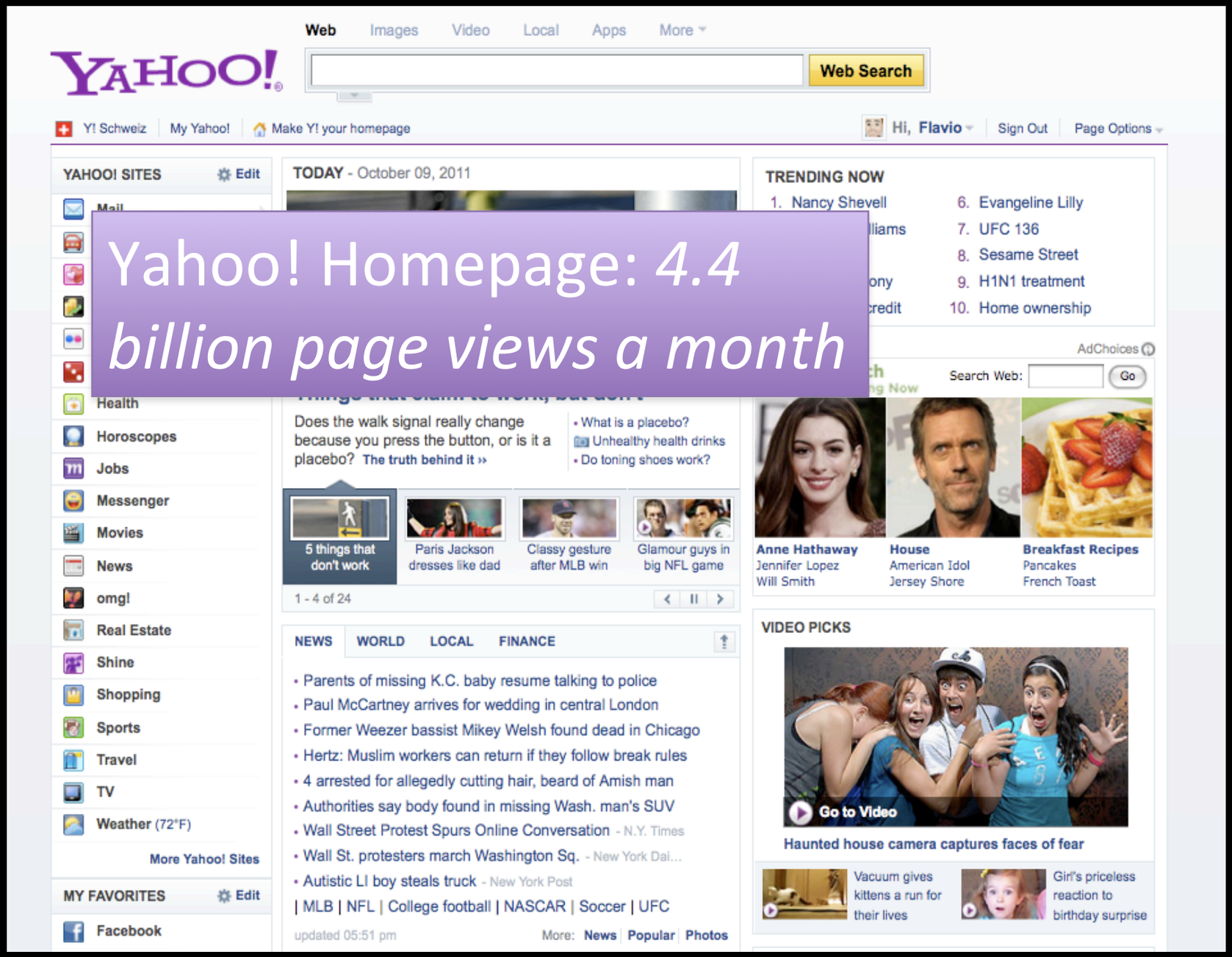
News

Yahoo!: Workload generated

Yahoo!: Workload generated

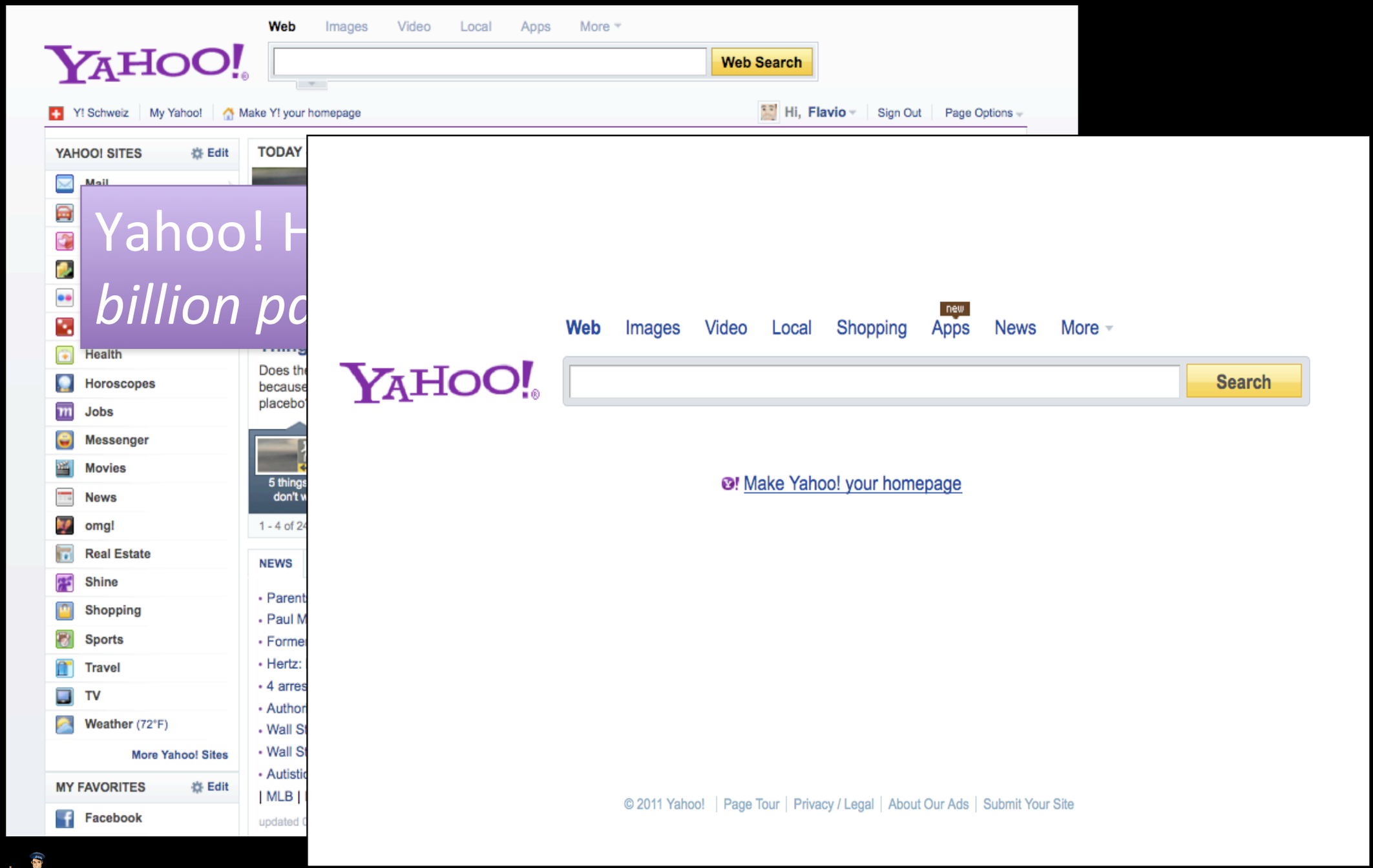


Yahoo!: Workload generated

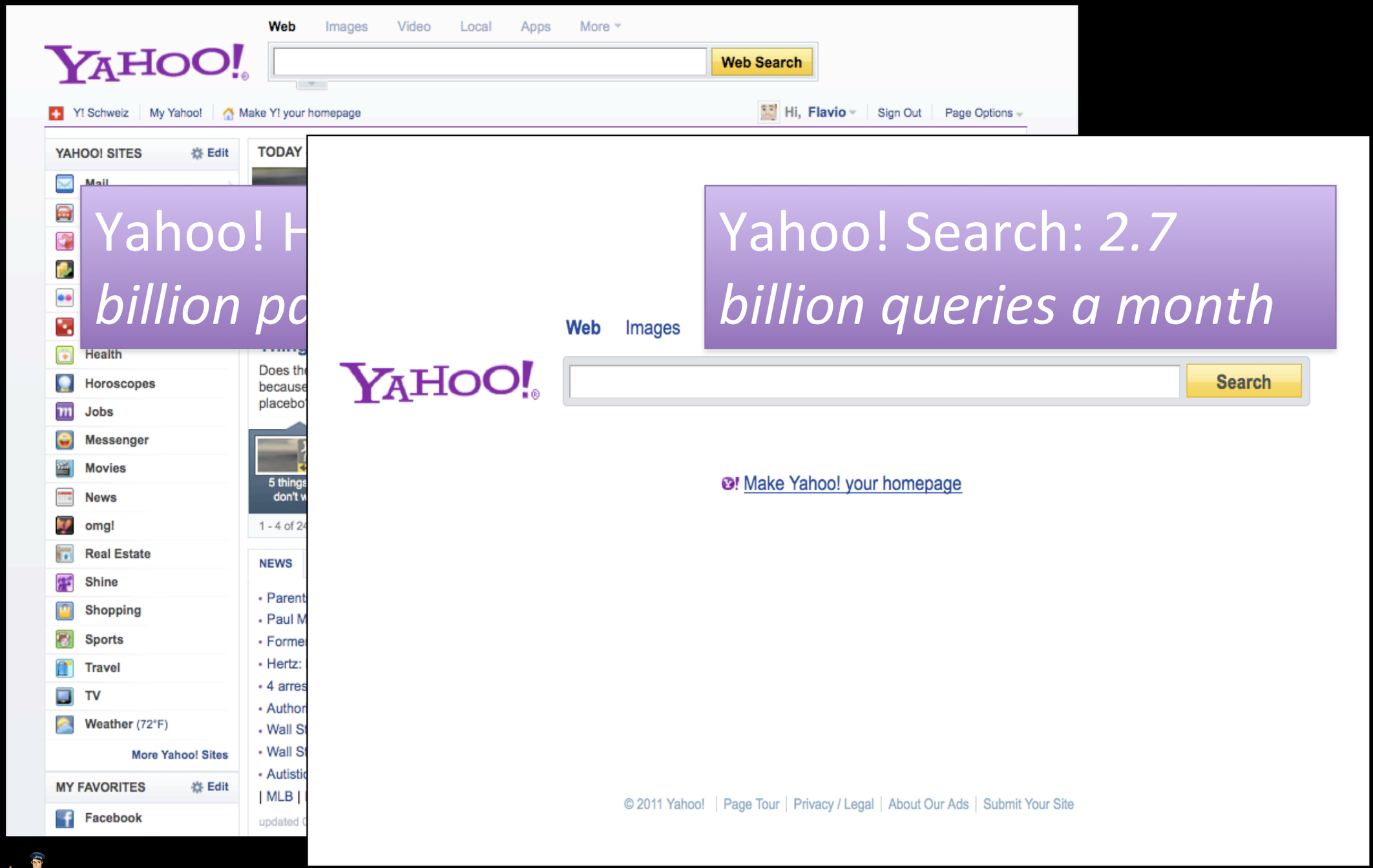


Yahoo! Homepage: *4.4 billion page views a month*

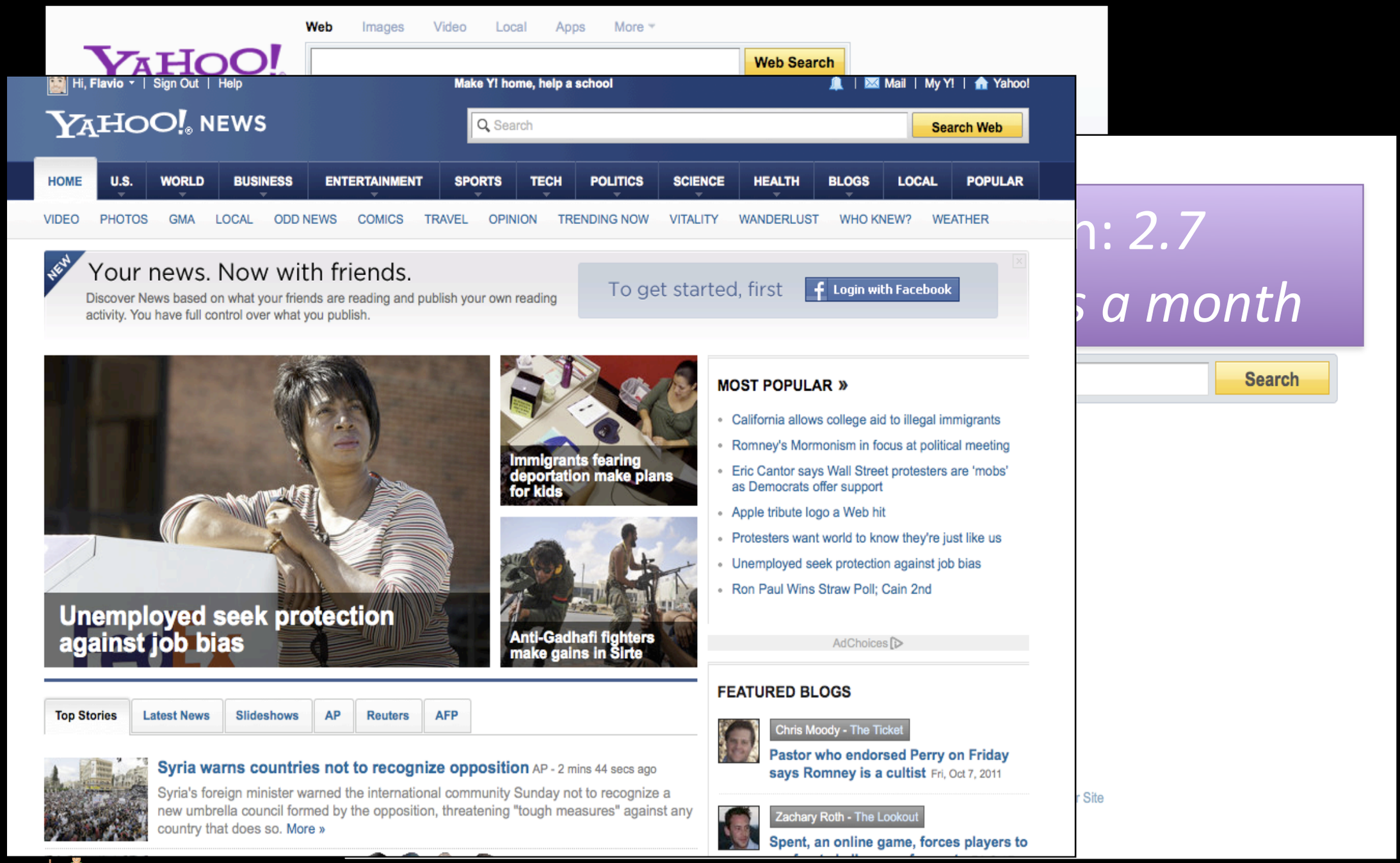
Yahoo!: Workload generated



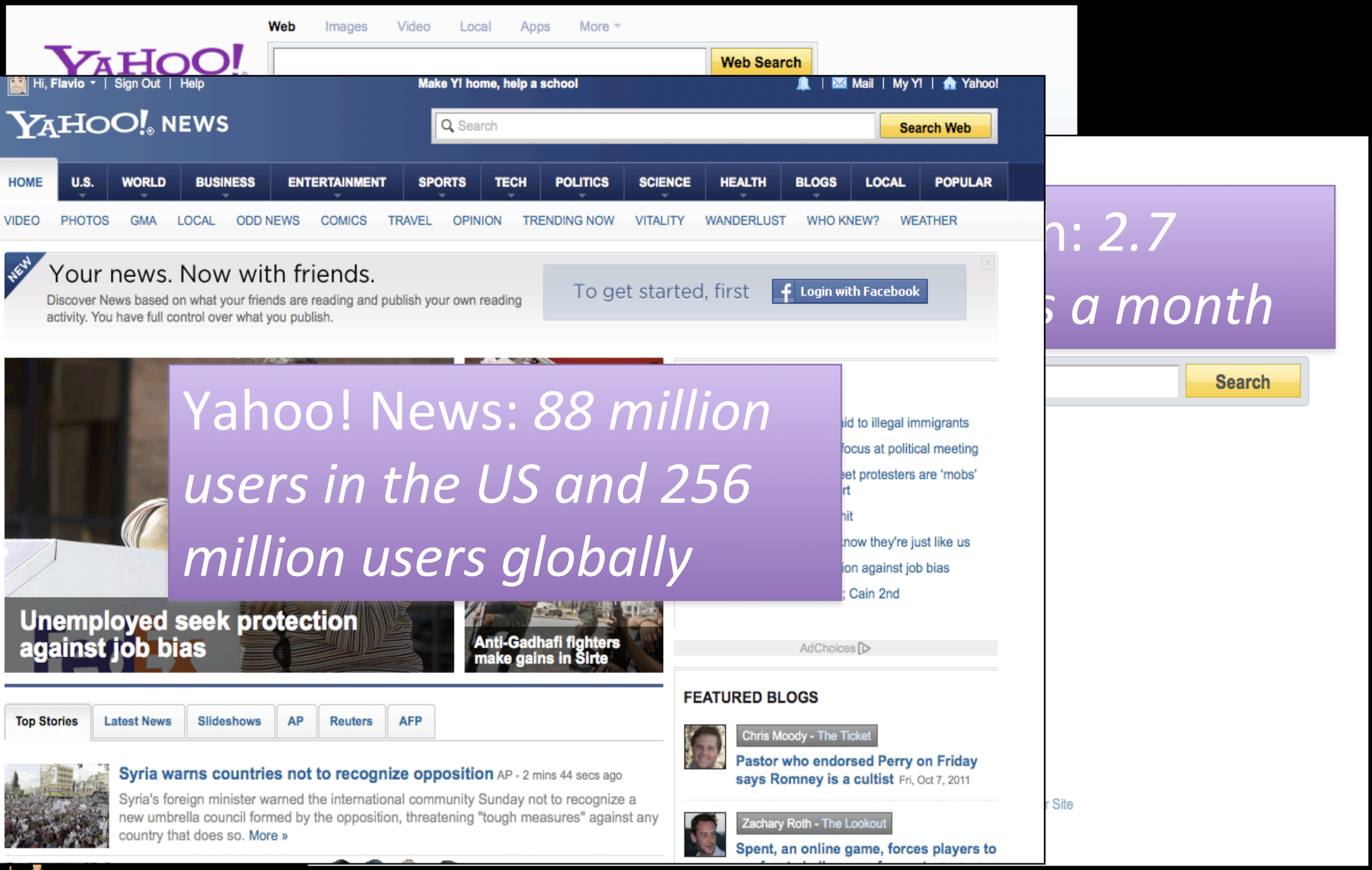
Yahoo!: Workload generated



Yahoo!: Workload generated



Yahoo!: Workload generated



Yahoo! Infrastructure

- Lots of servers
- Lots of processes
- High volumes of data
- Highly complex systems
- ... and developers are mere mortals

Yahoo! Lockport Data Center

by amusingplanet via Flickr

Copyright by Peter E. Lee via Flickr

Copyright by Shamus O’Reilly via Flickr

Copyright by Shamus O’Reilly via Flickr

... and in computer systems?

- Locks
- Queues
- Leader election
- Group membership
- Barriers
- Configuration

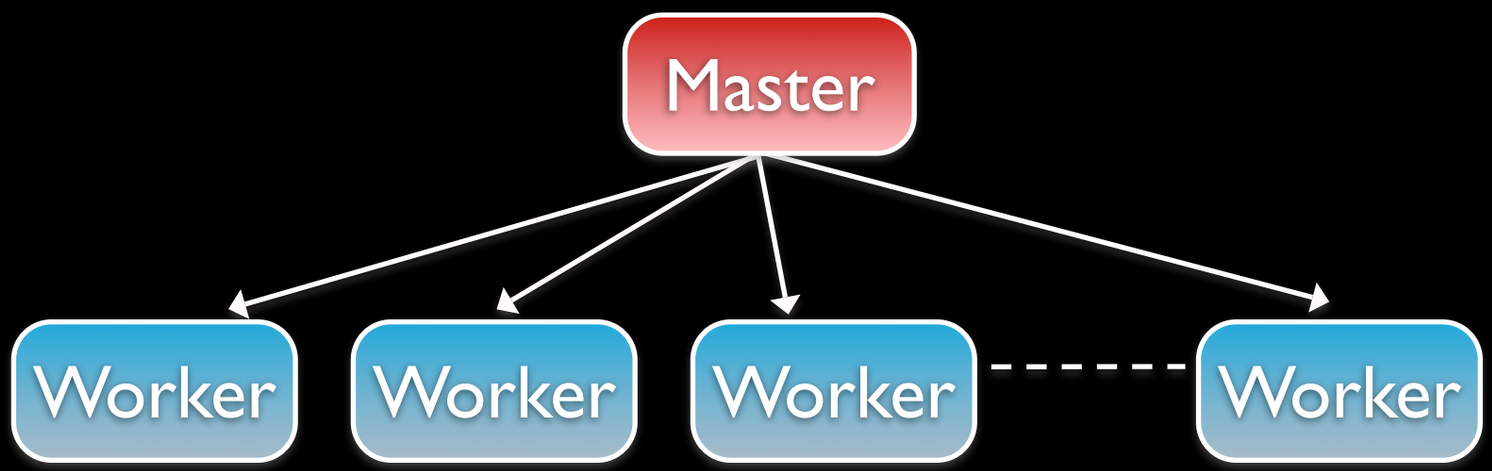
... and in computer systems?

- Locks
- Queues
- Leader election
- Group membership
- Barriers
- Configuration

Require knowledge of complex protocols

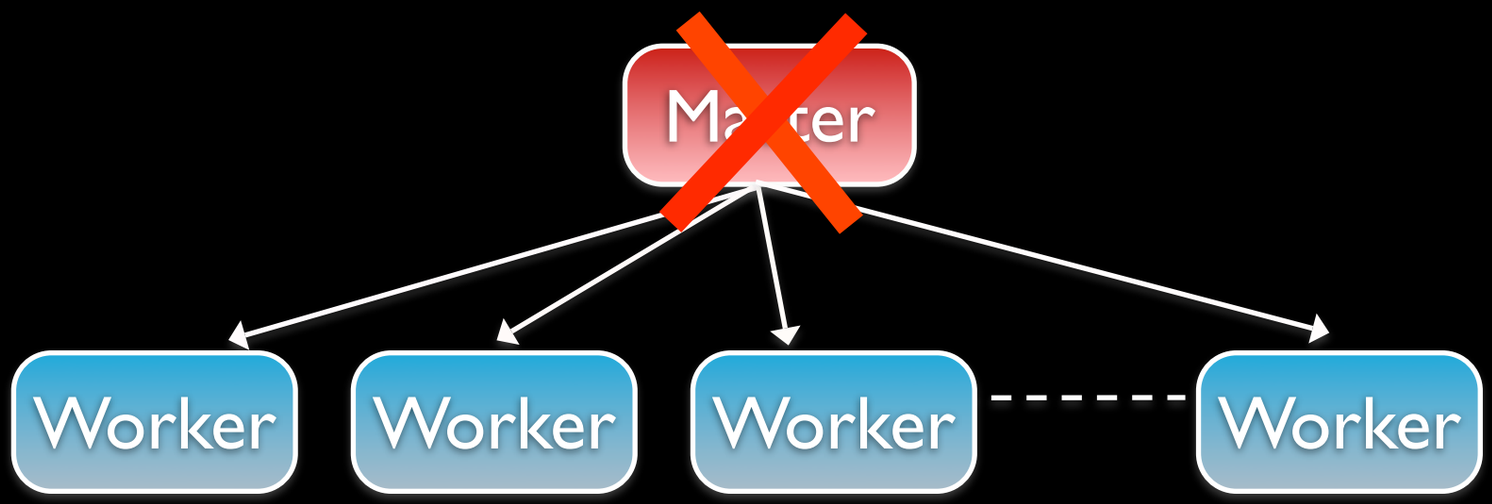
A simple model

- Work assignment
  - ✓ Master assigns work
  - ✓ Worker executes task assigned by master



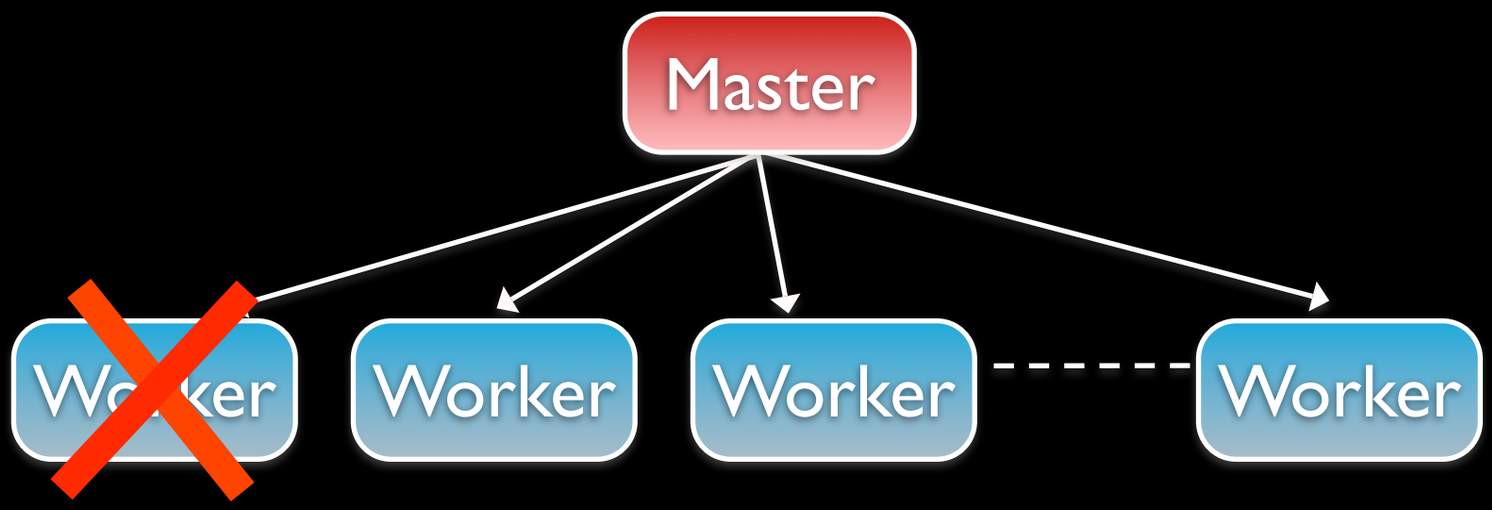
Master crashes

- ✓ Single point of failure
- ✓ No work assigned
- ✓ Need to select a new master



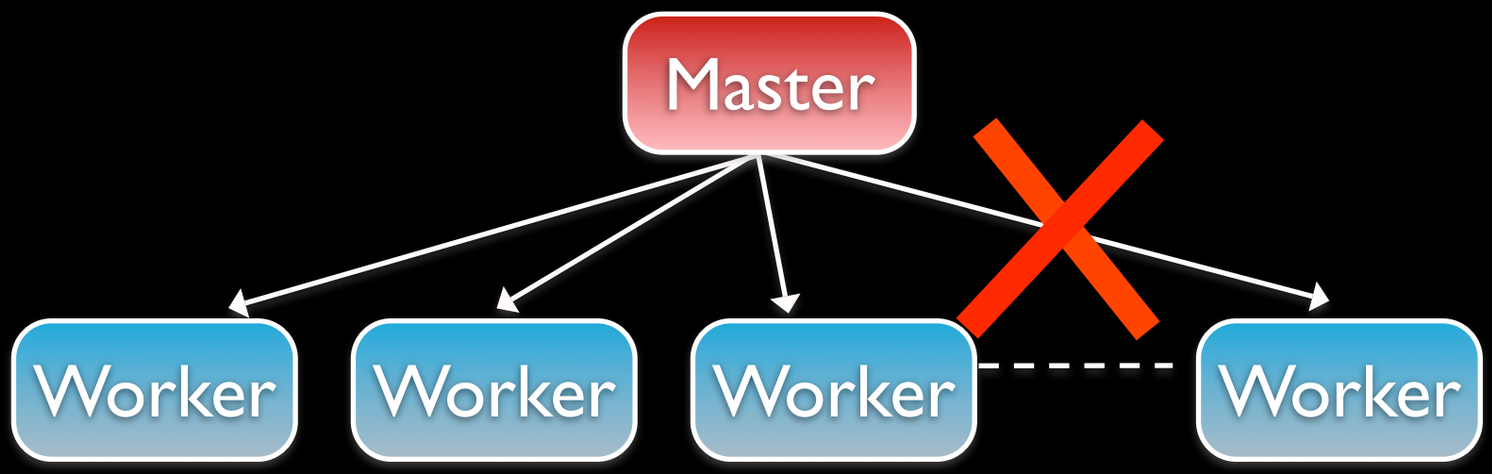
Worker crashes

- ✓ Not as bad... Overall system still works
- ✓ Some tasks won't be executed
- ✓ Need the ability to reassign tasks



Worker does receive assignment

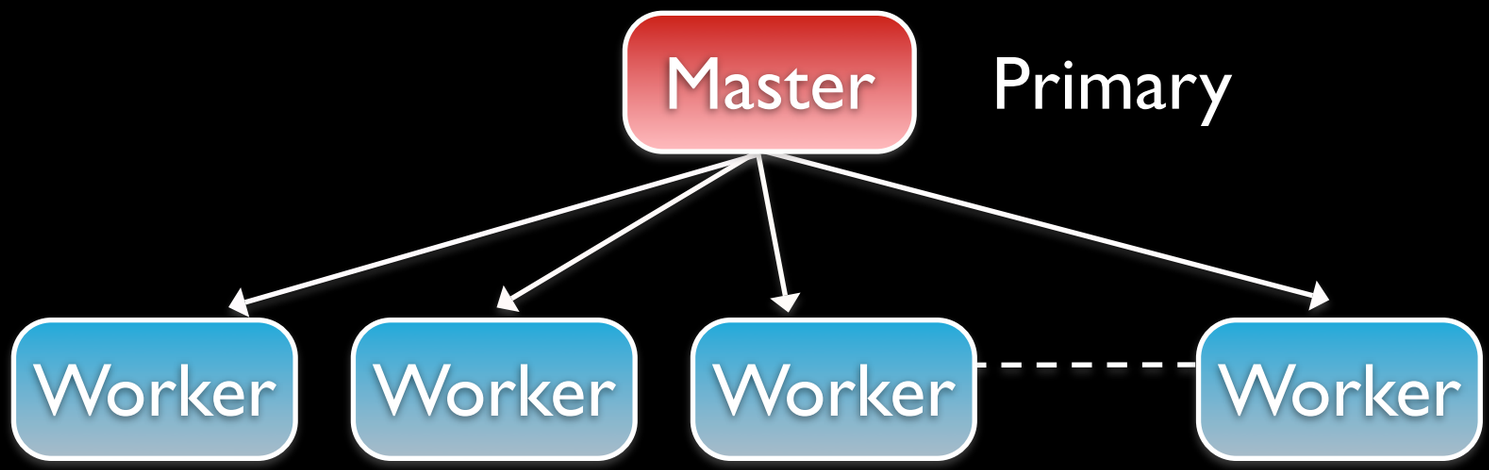
- ✓ Same problem, tasks don't get executed
- ✓ Need to guarantee that worker receives assignment



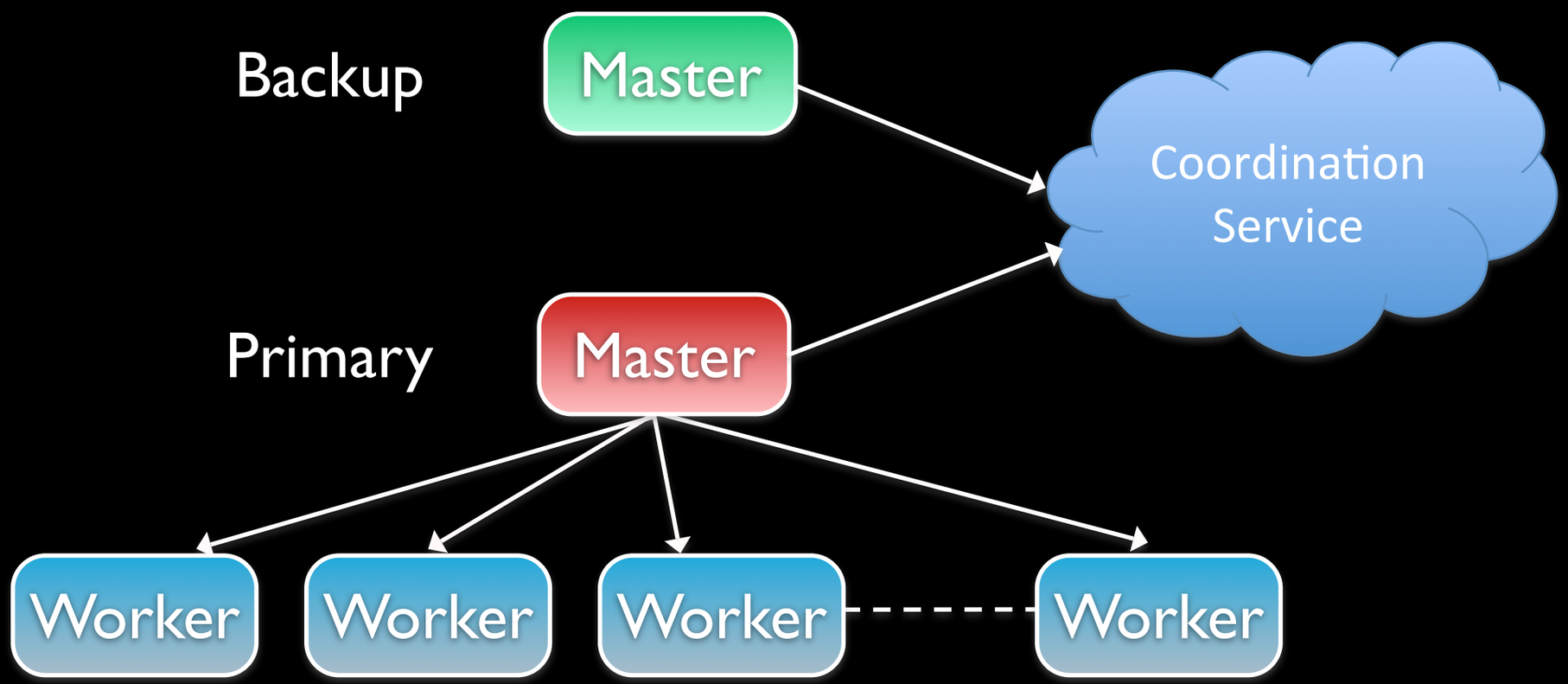
Fault-tolerant distributed system

What kinds of problems can we have here?

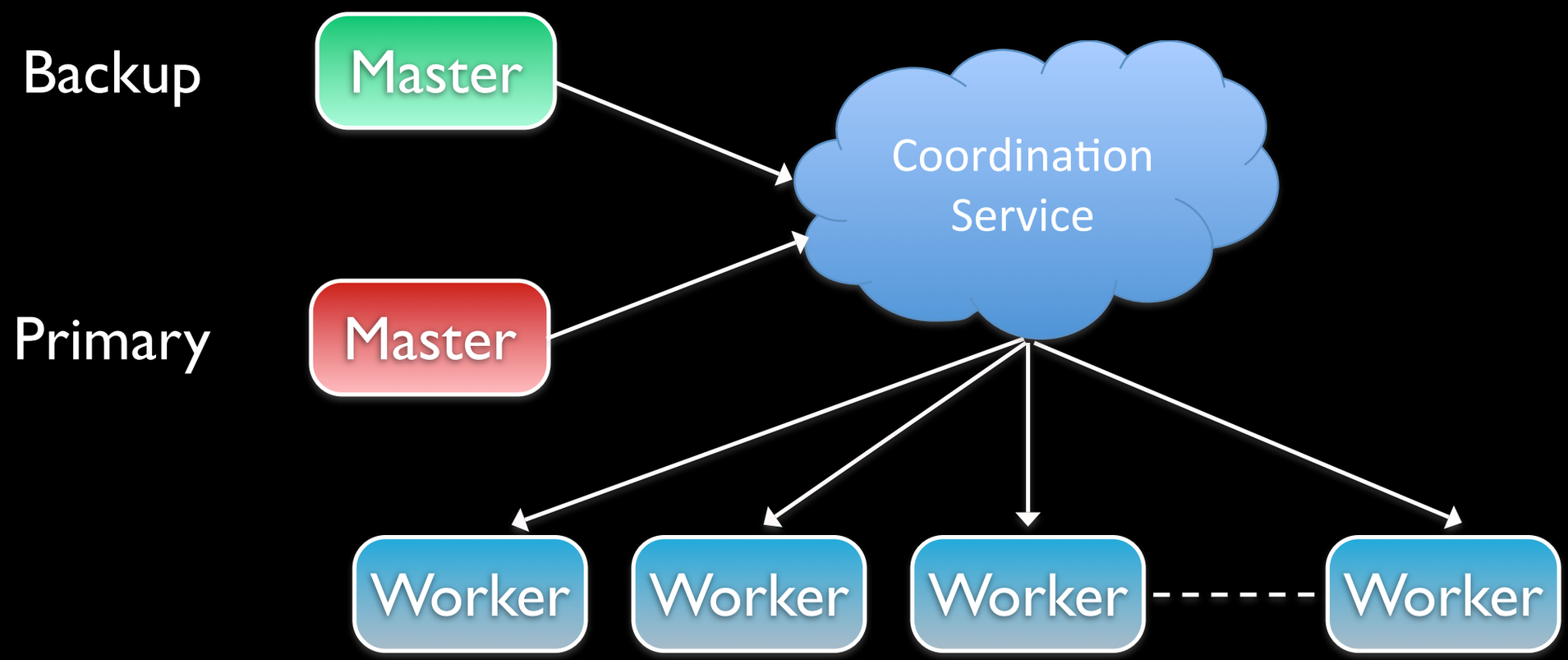
Backup



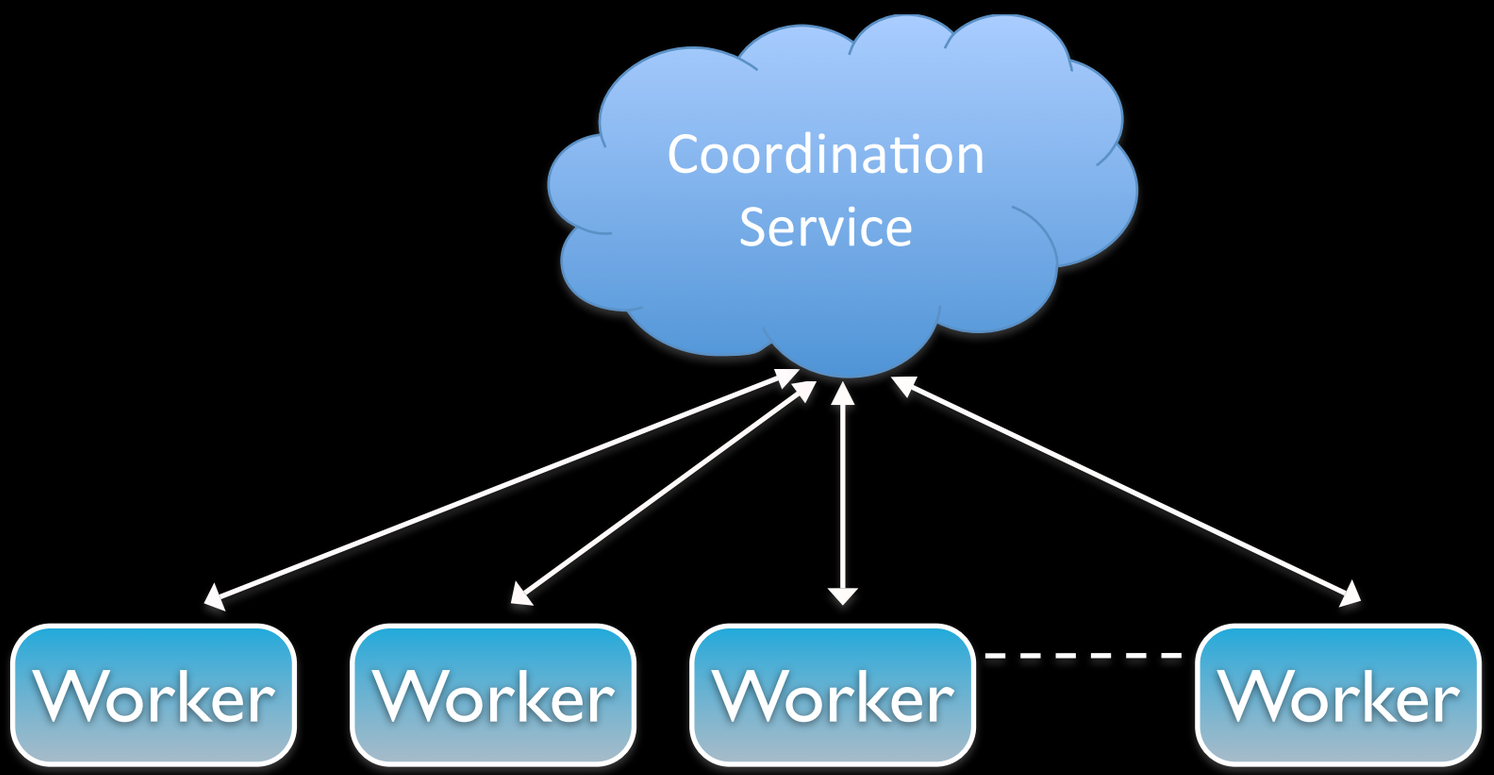
Fault-tolerant distributed system



Fault-tolerant distributed system



Fully distributed

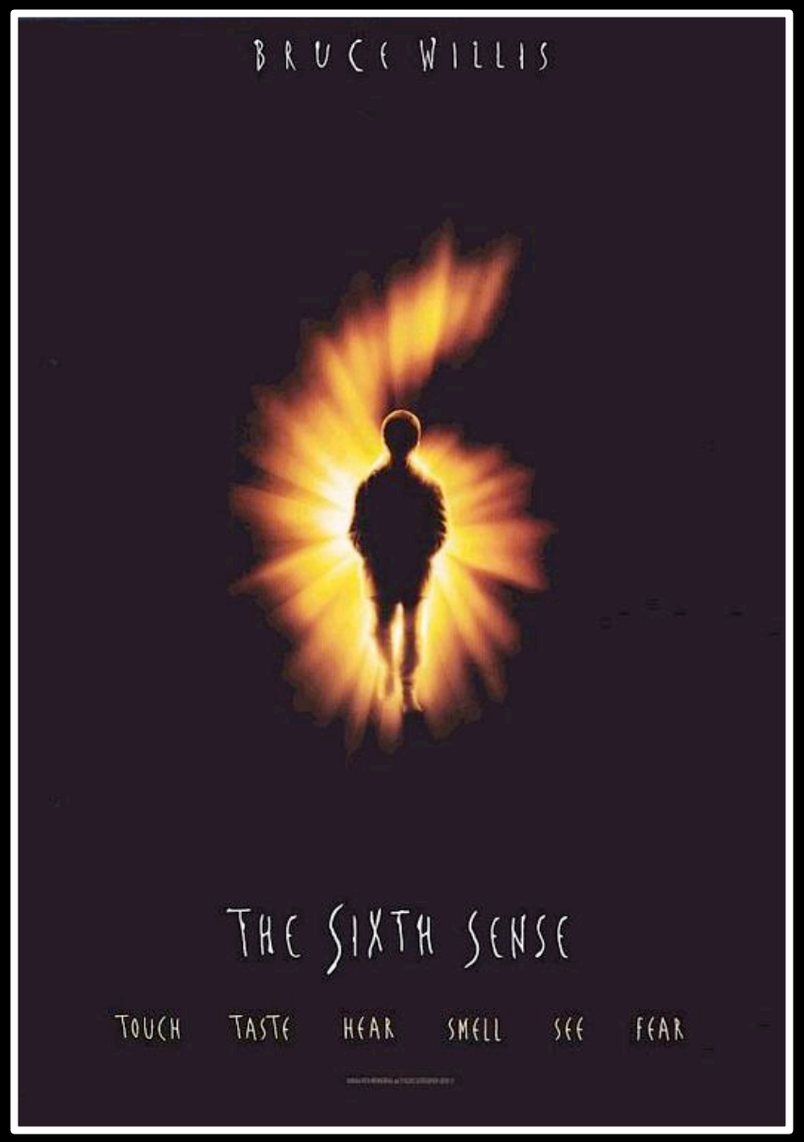


Fallacies of distributed computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. Network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. Network is homogeneous

Peter Deutsch, http://blogs.sun.com/jag/resource/Fallacies.html

One more fallacy



9. You know who is alive

Why is it difficult?

- FLP impossibility result
  - ✓ Asynchronous systems
  - ✓ Impossible is a single process can crash

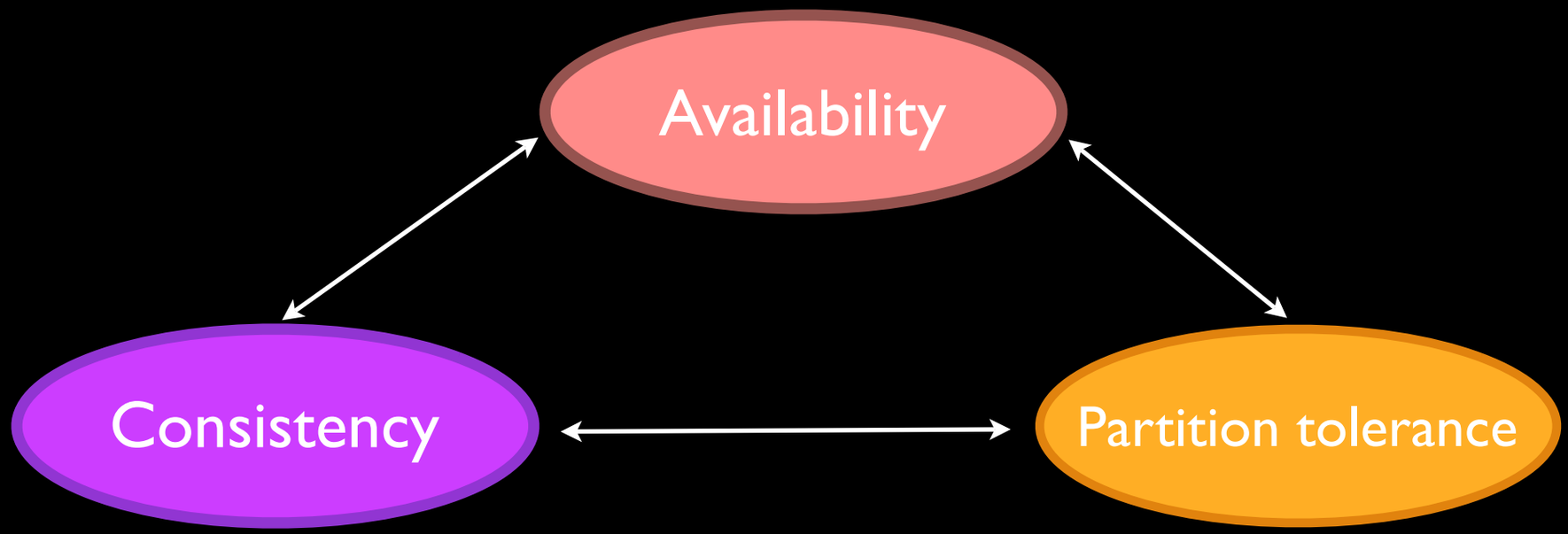
Fischer, Lynch, Paterson, ACM PODS, 1983

- According to Herlihy we do need consensus
  - ✓ Wait-free synchronization
  - ✓ Wait-free: Operations complete in a finite number of steps
  - ✓ Equivalent to solving consensus for n processes

Herlihy, ACM TOPLAS, 1991

Why is it difficult?

- CAP Principle
  - ✓ Can't have availability, consistency, and partition tolerance



The case for a coordination service

- Many fallacies to stumble upon
- Many impossibility results
- Several common requirements across applications
  - ✓ Duplicating is bad
  - ✓ Duplicating poorly is even worse
- Coordination service
  - ✓ Implement it once and well

Current systems

- Google Chubby
  - ✓ Lock service

Burrows, USENIX OSDI, 2006

- Microsoft Centrifuge
  - ✓ Lease service

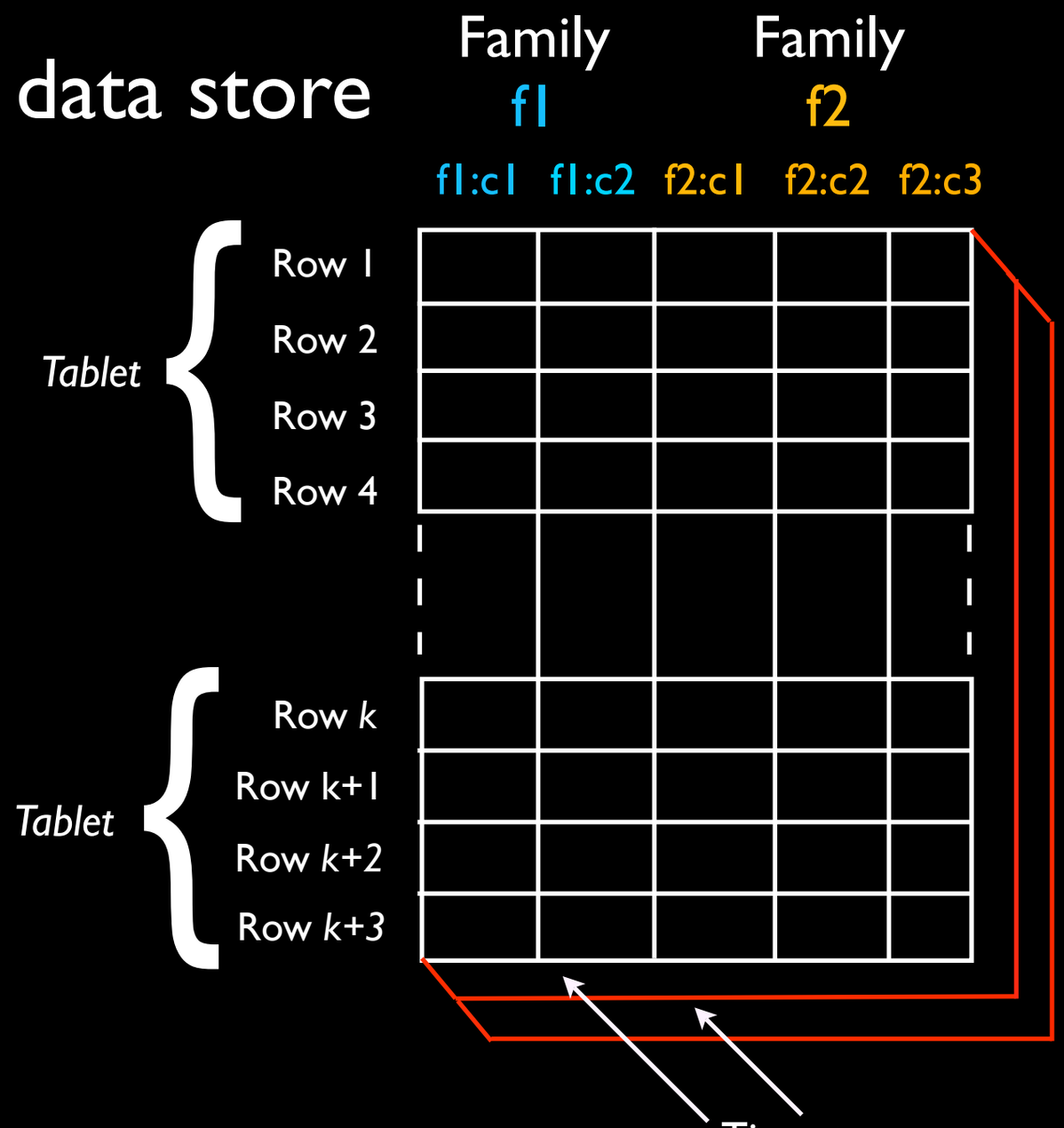
Adya et al., USENIX NSDI, 2010

- Apache ZooKeeper
  - ✓ Coordination kernel
  - ✓ Initially contributed by Yahoo!

Hunt et al., USENIX ATC, 2010

Example - Bigtable, Hbase

- Sparse column-oriented data store
  - ✓ Tablet: Range of rows
  - ✓ Unit of distribution
- Architecture
  - ✓ Master
  - ✓ Tablet servers

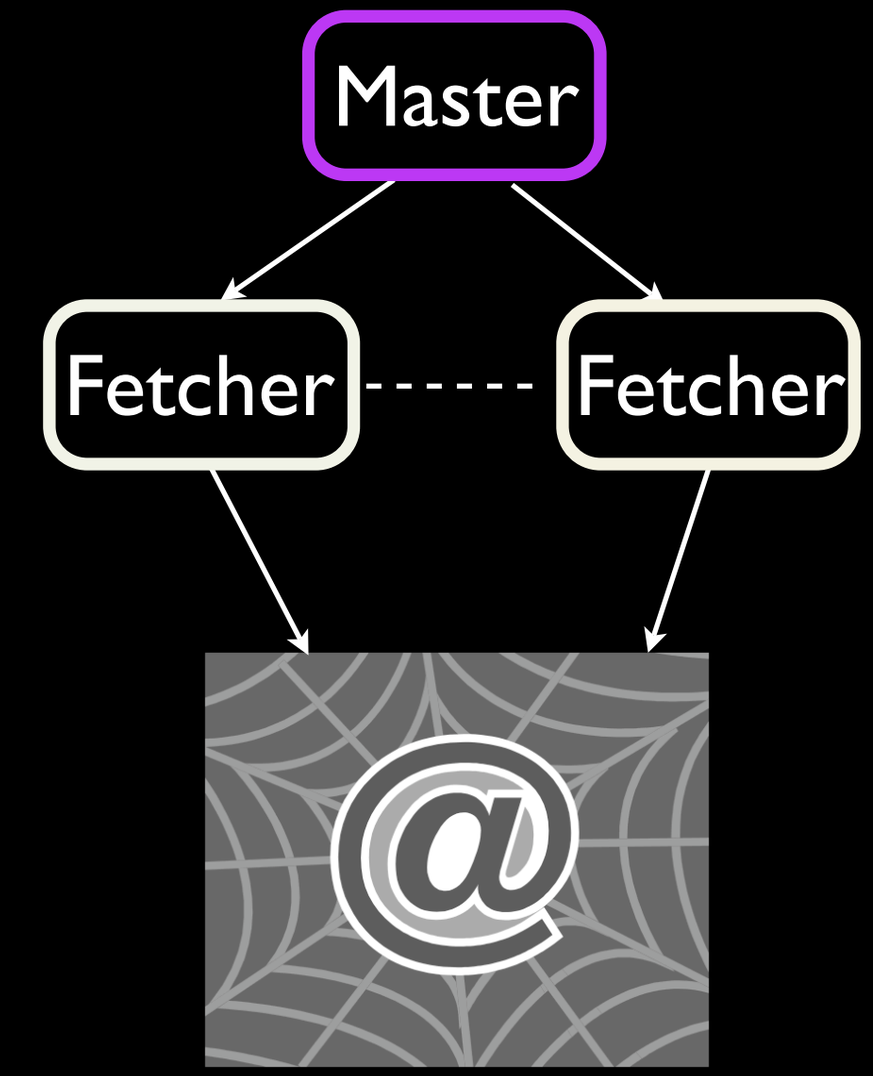


Example - Bigtable, Hbase

- Master election
  - ✓ Master crashes
- Metadata management
  - ✓ ACLs, Tablet metadata
- Rendezvous
  - ✓ Find tablet server
- Live tablet servers

Example - Web crawling

- Fetching service
  - ✓ Fetch web pages for search engines
- Master election
  - ✓ Assign work
- Metadata management
  - ✓ Politeness constraints
  - ✓ Shards
- Live workers



Example - Kafka, Pub/Sub

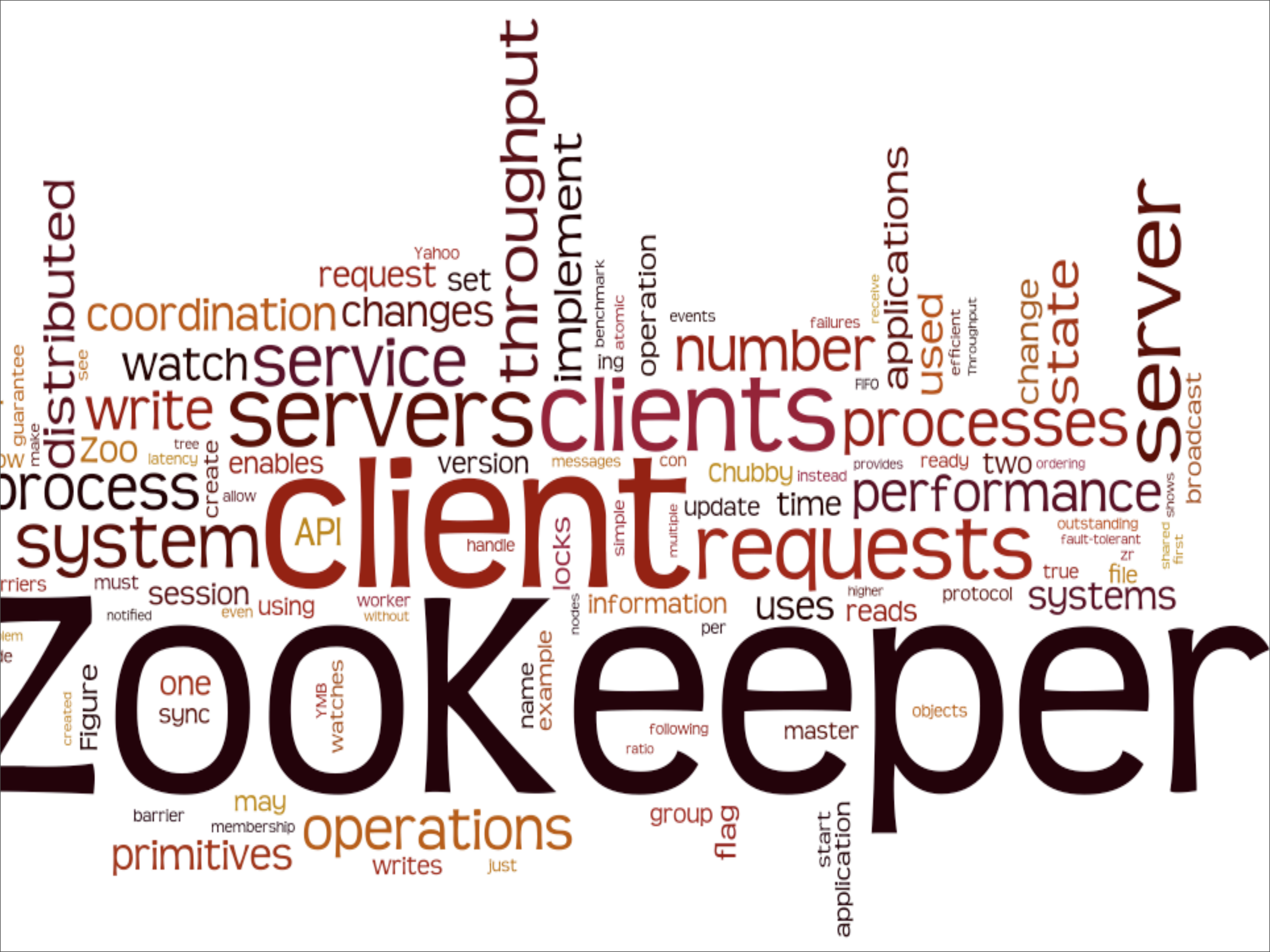
- Based on topics
- Topics are partitioned across brokers
- Consumer groups
  - ✓ Multiple consumers for a topic
- Coordination requirements
  - ✓ Metadata
    - ➡ Each message is consumed by a single consumer
    - ➡ Each partition is owned by a consumer
  - ✓ Crash detection

And more examples...

- Google File System
  - ✓ Master election
  - ✓ File system metadata
- Hedwig - Pub/Sub
  - ✓ Topic metadata
  - ✓ Topic assignment
  - ✓ Elasticity

At Yahoo!...

- Has been used for:
  - ✓ Fetching service
  - ✓ Manage workflows in Hadoop (e.g., feed ingestion)
  - ✓ Content optimization (distributed services)
  - ✓ ...
- Largest cluster I'm aware of
  - ✓ Around 5,000 - 10,000 clients


ZooKeeper
client
servers
clients
processes
server
requests
process
system
performance
service
throughput
implement
distributed
applications
number
state
write
watch
coordination
changes
request
set
Yahoo
operation
benchmark
atomic
ing
events
failures
receive
used
efficient
Throughput
change
FIFO
broadcast
version
enables
Zoo
tree
latency
create
messages
con
Chubby
instead
provides
ready
two
ordering
guarantee
see
make
allow
API
handle
locks
simple
multiple
update
time
outstanding
fault-tolerant
zr
shared
first
shows
true
file
must
session
using
even
notified
worker
without
information
nodes
uses
reads
higher
per
protocol
systems
one
sync
Figure
created
YMB
watches
name
example
following
ratio
master
objects
may
barrier
membership
operations
primitives
writes
just
group
flag
start
application

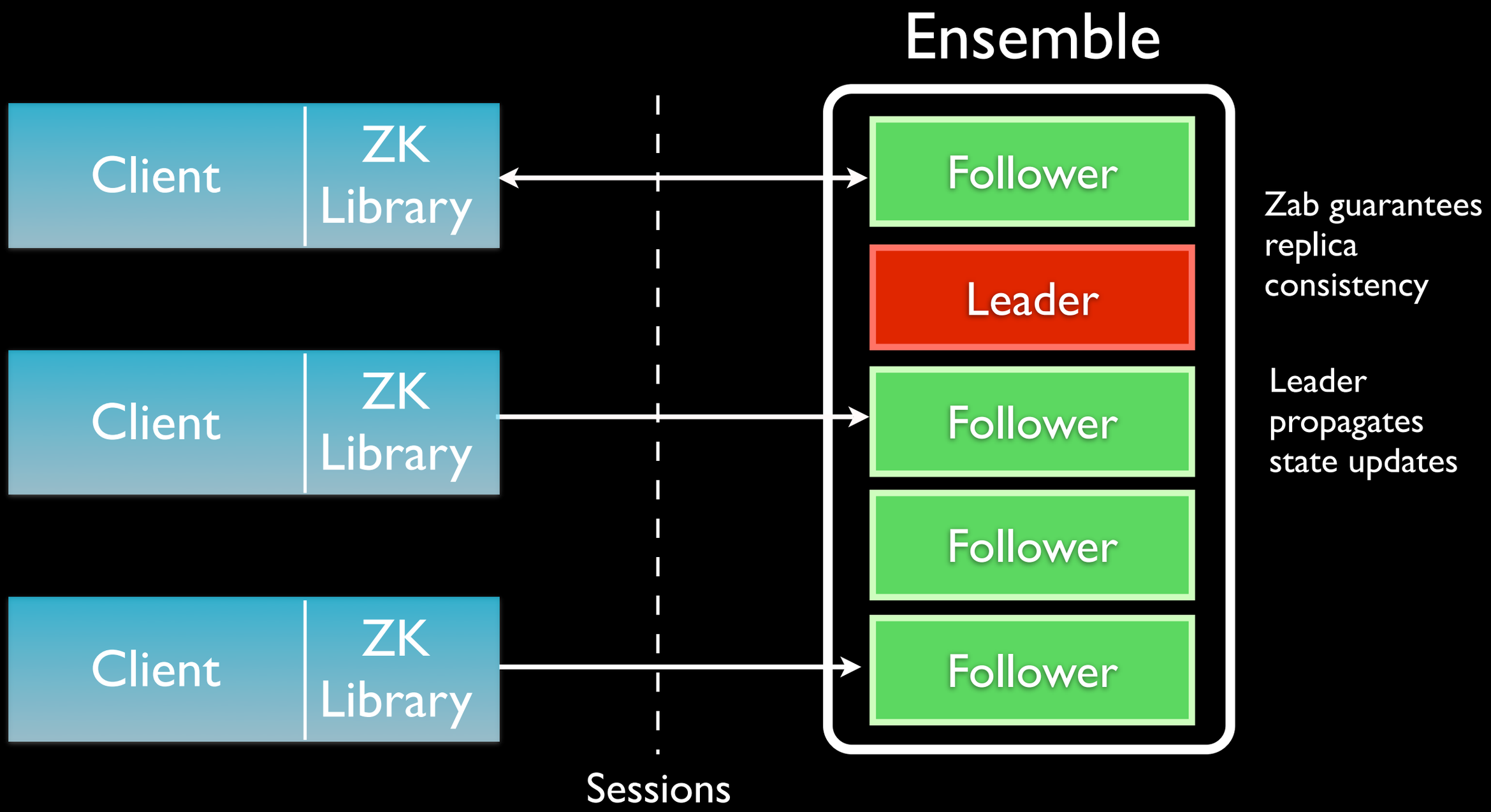
ZooKeeper introduction

- Coordination kernel
  - ✓ Does not export concrete primitives
  - ✓ Recipes to implement primitives
- File-system based API
  - ✓ Manipulates small data nodes: *znodes*
  - ✓ State is a hierarchy of znodes

More introduction

- Stores database in memory
  - ✓ Handles high load
  - ✓ Handy when communicating with a large number of processes
- Single data-center applications, originally
  - ✓ Some cases of cross-colo deployments

ZooKeeper: Design



What's difficult here?

- Electing a leader
  - ✓ All live processes are potential leaders
  - ✓ Communication pattern is arbitrary
- Replicating the state
  - ✓ Zab: a high-performance broadcast protocol
  - ✓ Enables multiple outstanding operations

What do clients see?

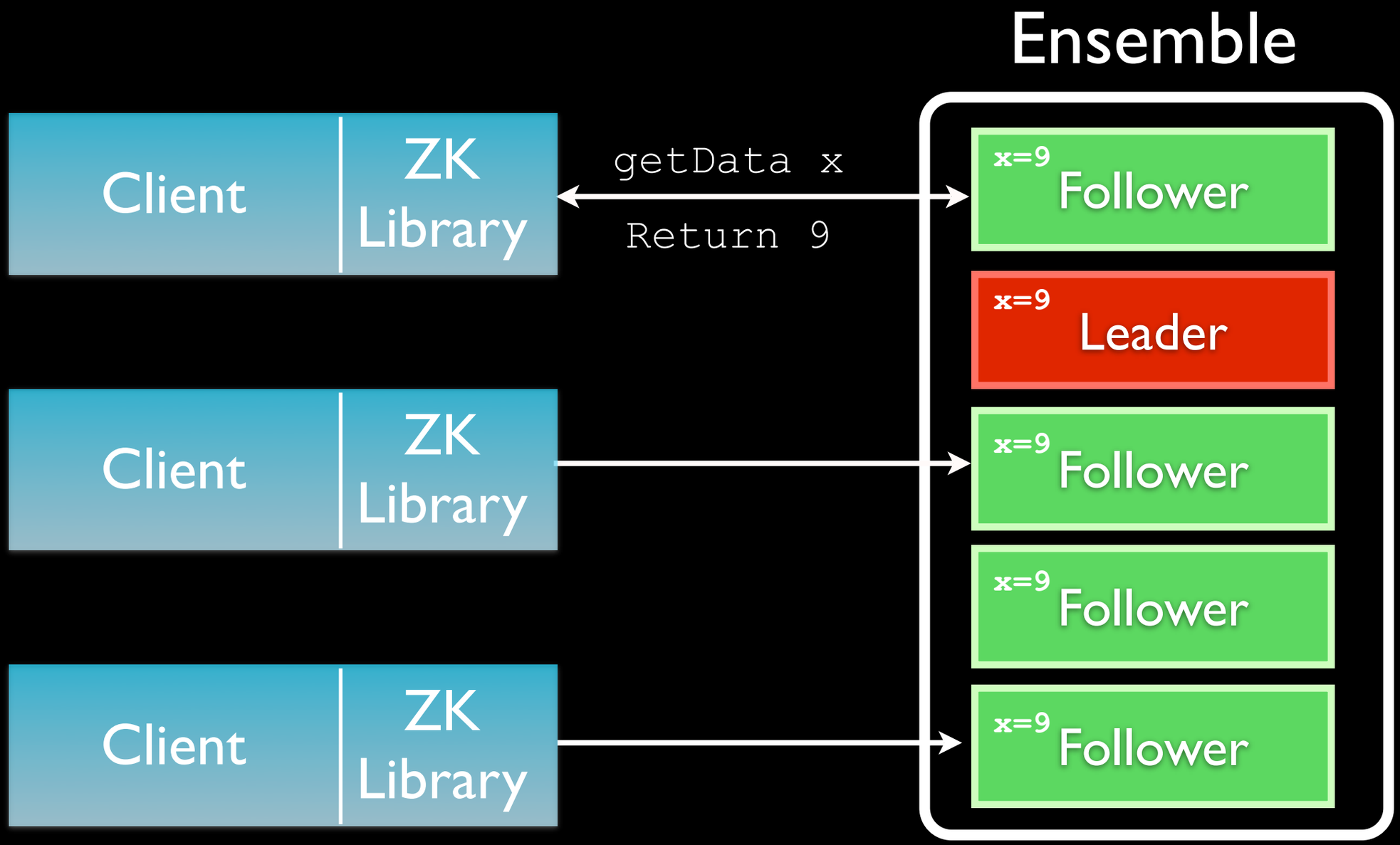
- Semantics of sessions
- Prefix of operations are executed

- Upon a disconnection
  - ✓ A server tries to contact another server
  - ✓ Before session expires: connect to new server
  - ✓ Server must have seen a higher transaction id

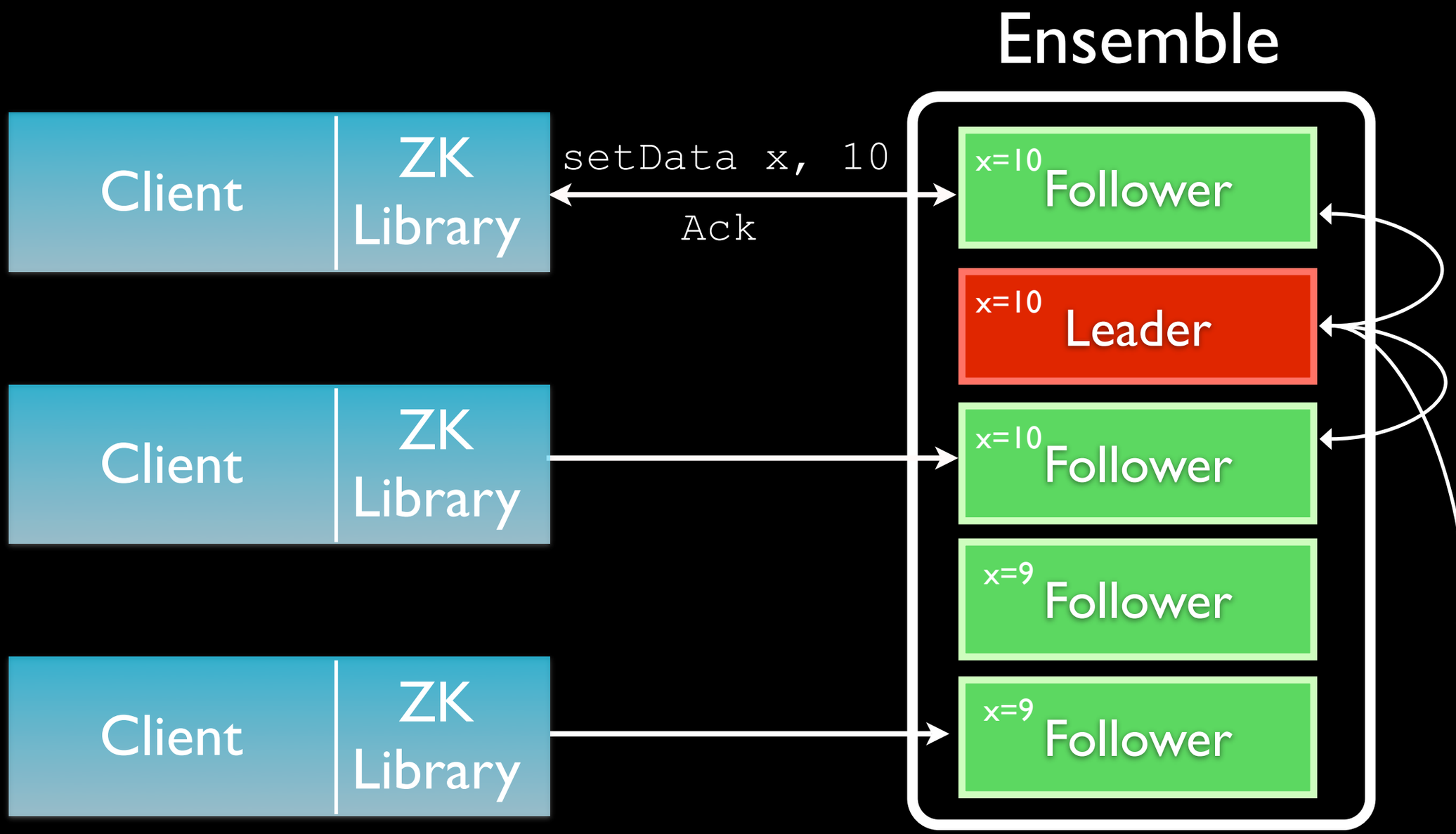
ZooKeeper API

- Create znodes: `create`
  - ✓ Persistent, ephemeral, sequential
- Read and modify data: `getData, setData`
- Read the children of znode: `getChildren`
- Check if znode exists: `exists`
- Delete znode: `delete`

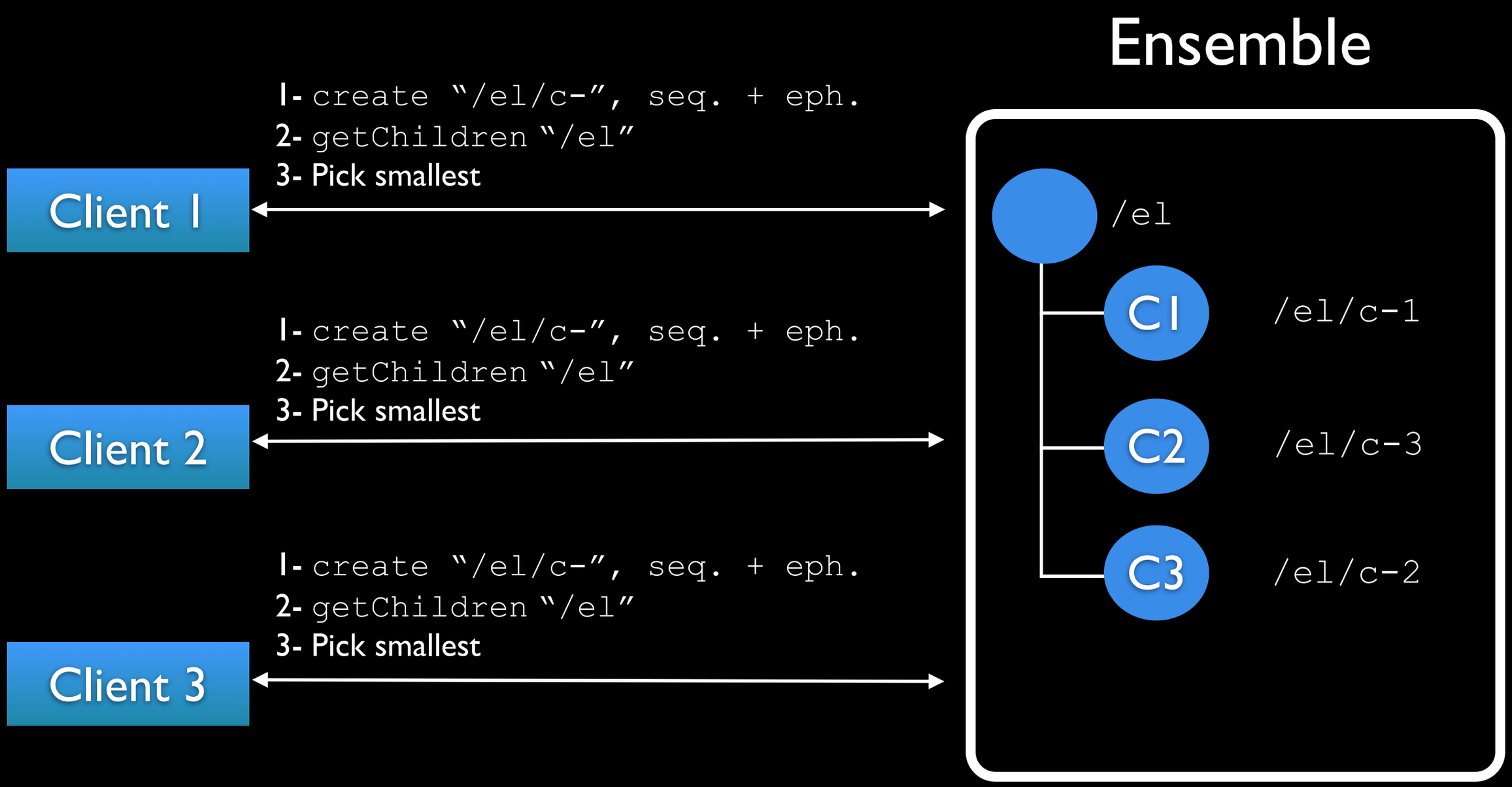
ZooKeeper: Reads



ZooKeeper: Writes



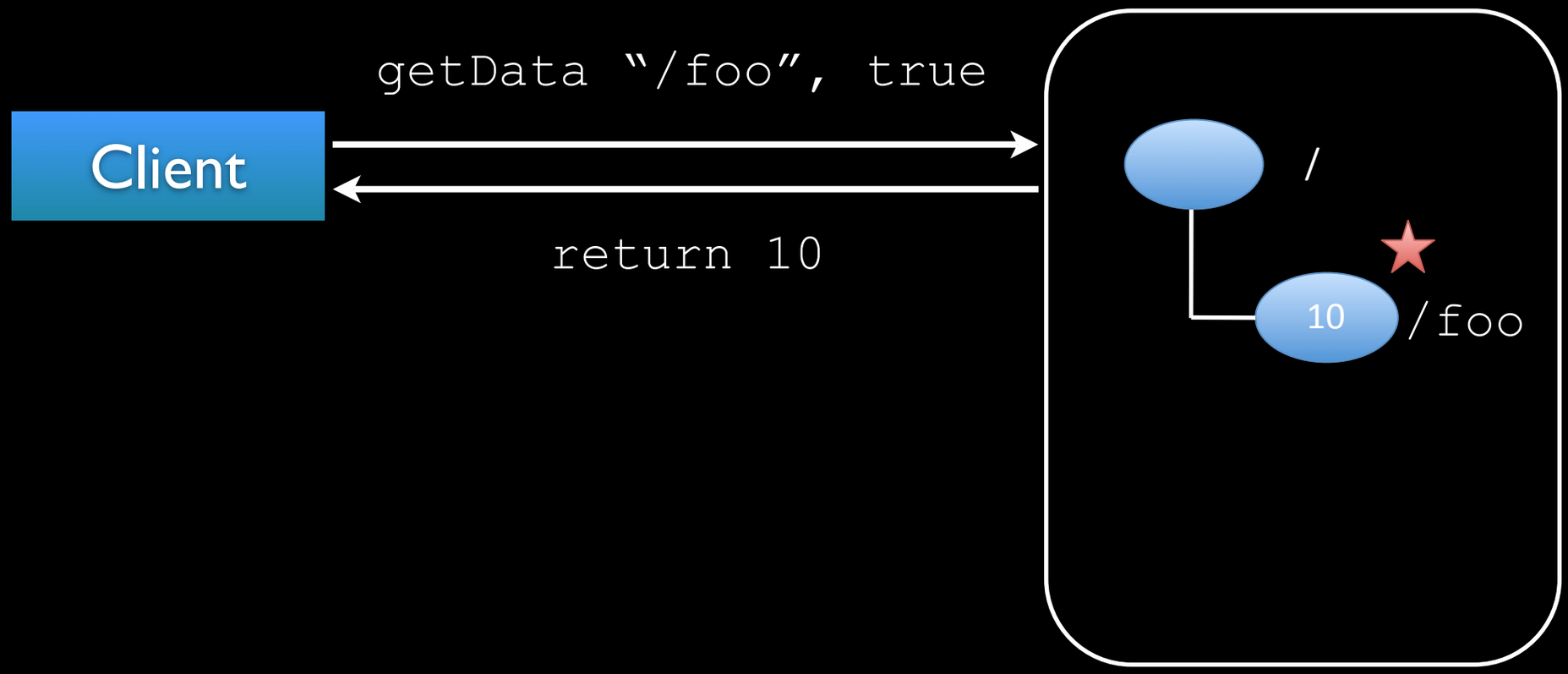
Example


Ensemble
1- create "/el/c-", seq. + eph.
2- getChildren "/el"
3- Pick smallest
Client 1
1- create "/el/c-", seq. + eph.
2- getChildren "/el"
3- Pick smallest
Client 2
1- create "/el/c-", seq. + eph.
2- getChildren "/el"
3- Pick smallest
Client 3
/el
C1
/el/c-1
C2
/el/c-3
C3
/el/c-2

Znode changes

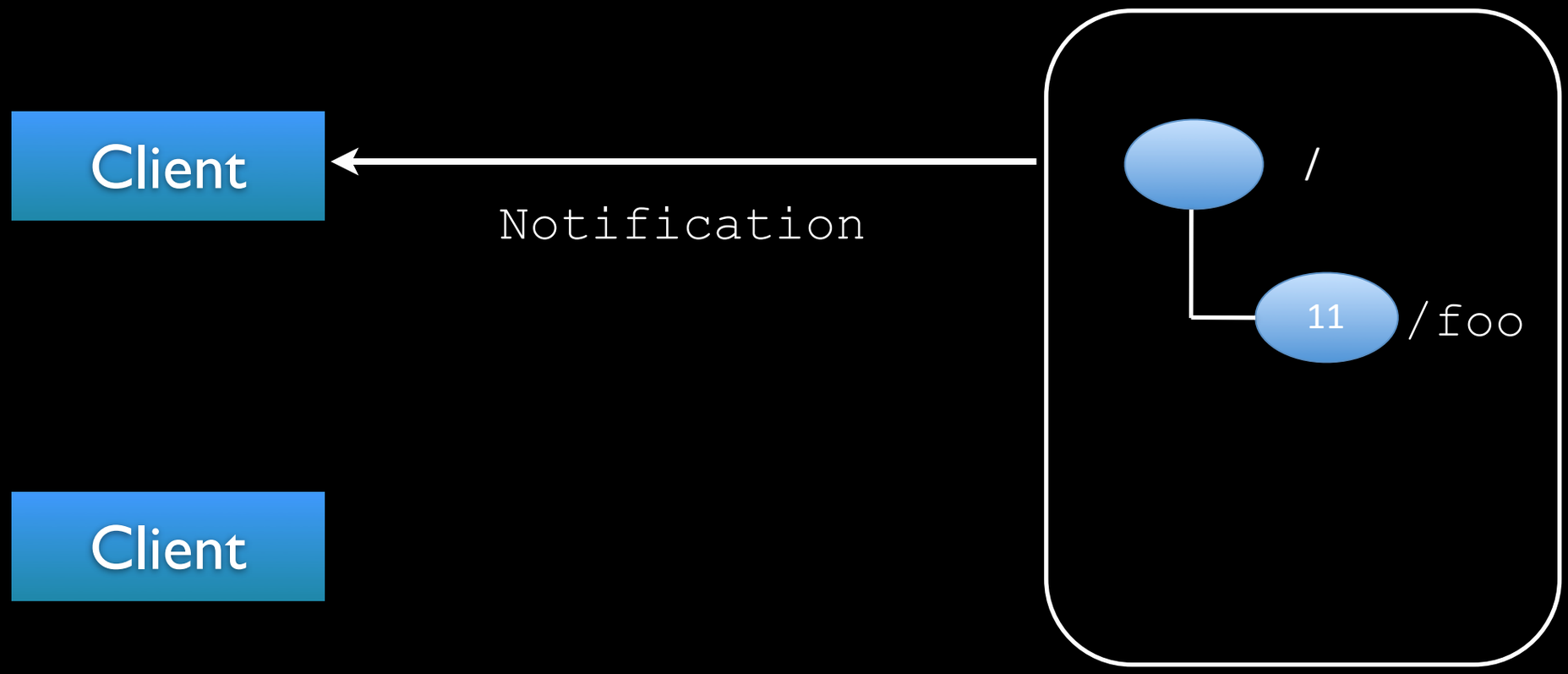
- Znode changes
  - ✓ Data is set
  - ✓ Node is created or deleted
  - ✓ *Etc…*
- Learn of znode changes
  - ✓ Set a *watch*
  - ✓ Upon change, client receives a *notification* before new updates

Watches



Watches

Watches

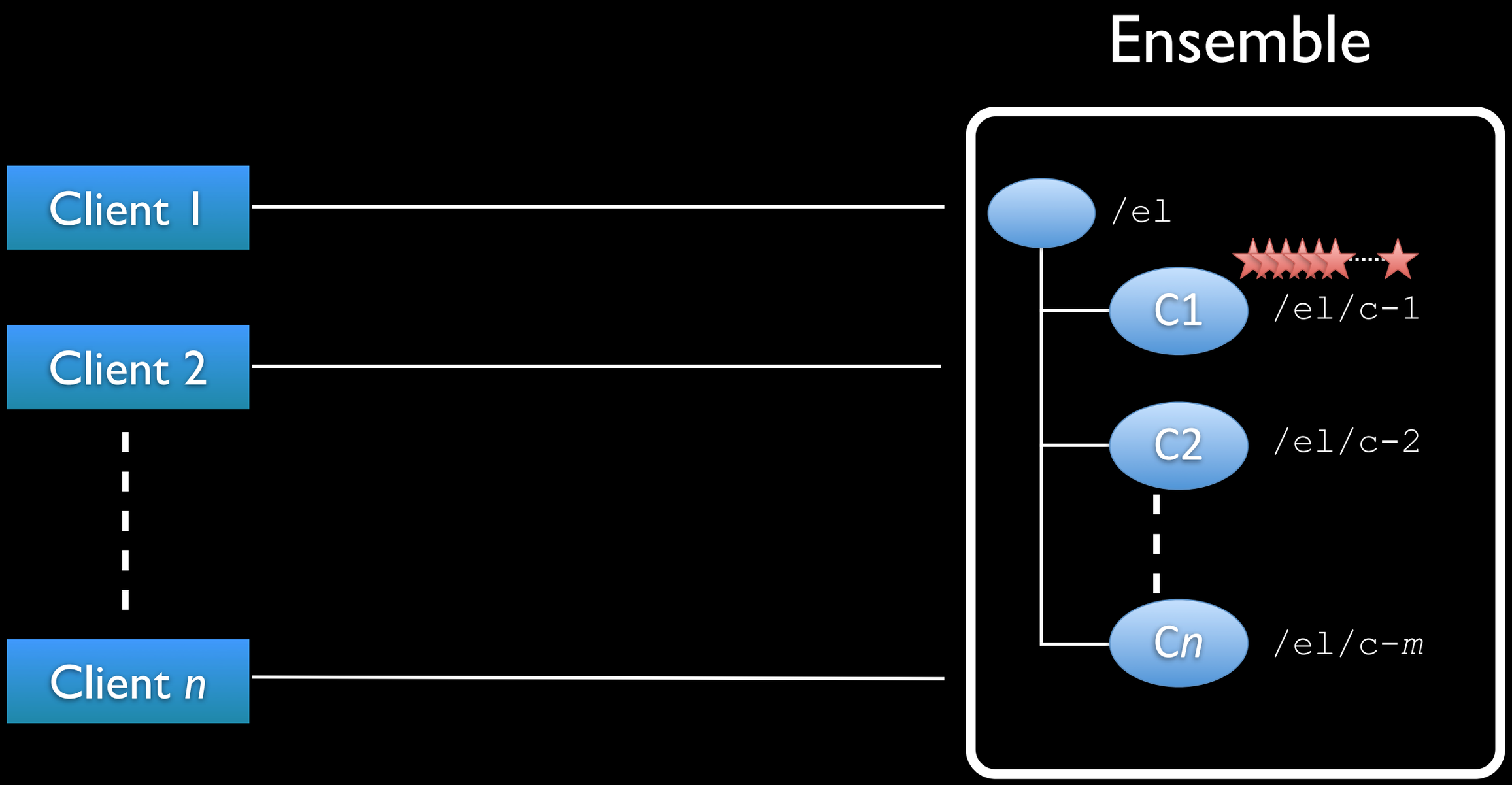


Client

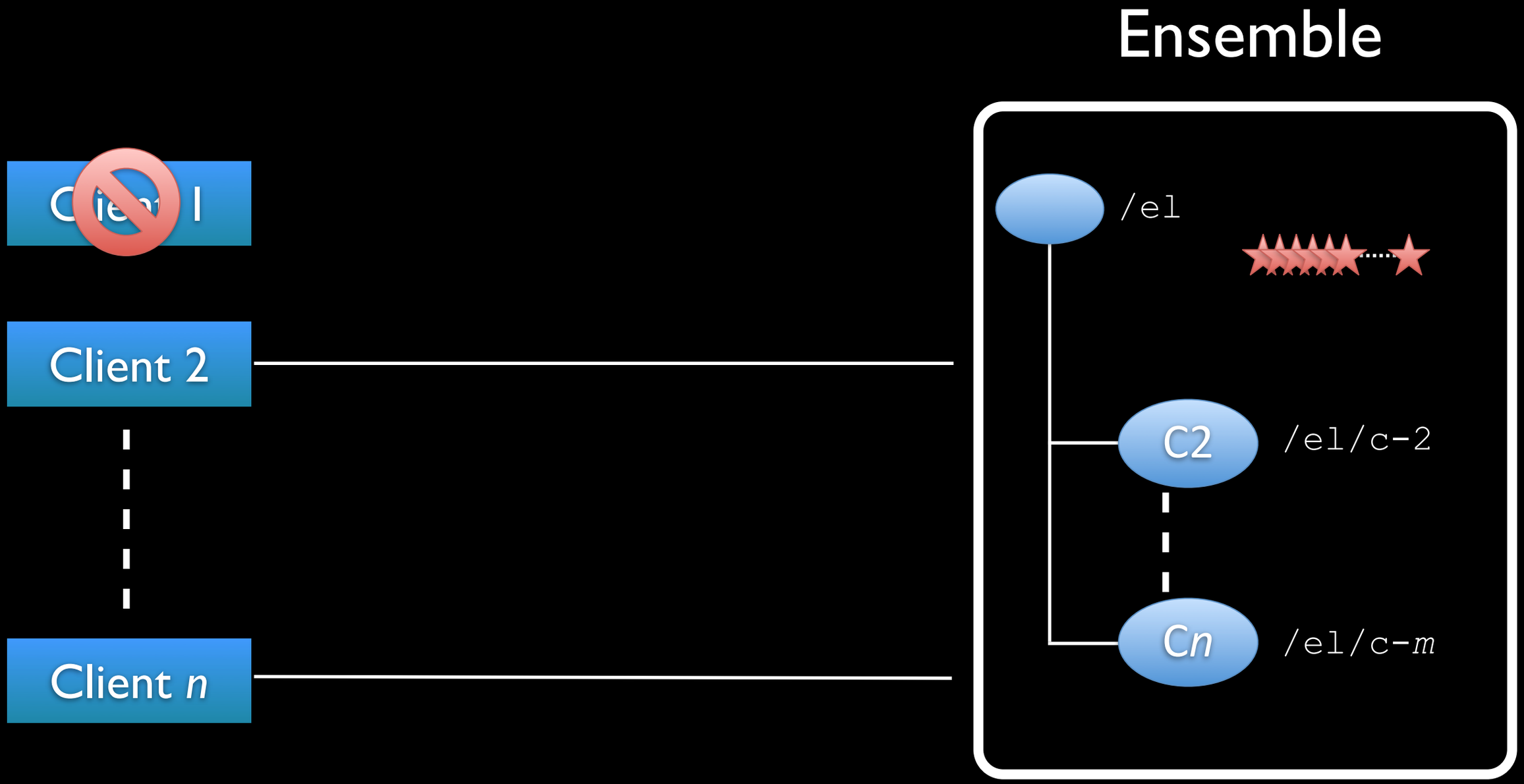
Watches, Locks, and the herd effect

- Herd effect
  - ✓ Large number of clients wake up simultaneously
- Undesirable effect
  - ✓ Load spikes

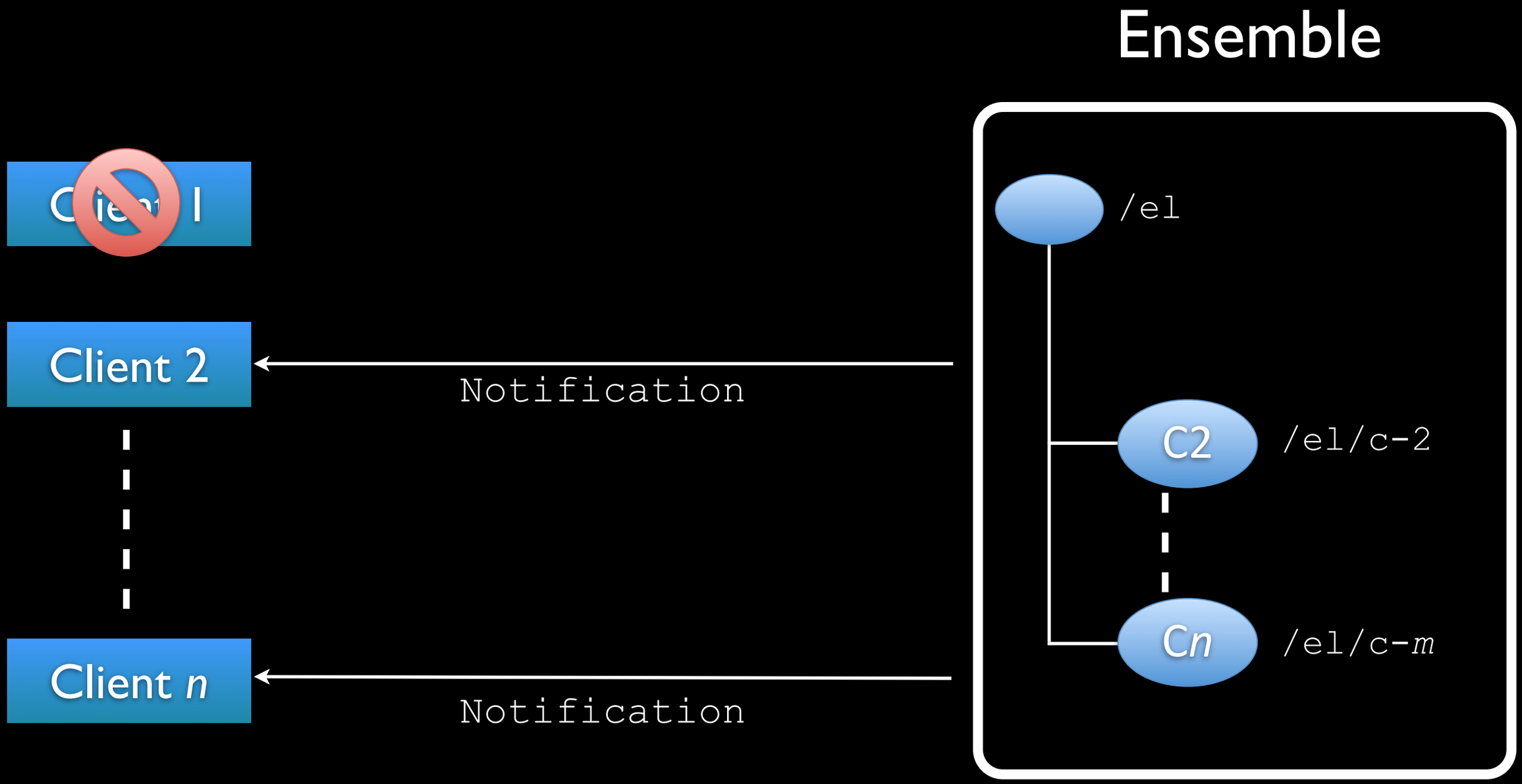
Watches, Locks, and the herd effect



Watches, Locks, and the herd effect



Watches, Locks, and the herd effect



Watches, Locks, and the herd effect

- A solution: Use order of clients

  ✓ Each client

  ➡ Pick znode z preceding its own znode in the sequential order

  ➡ Watch z

  ✓ A single notification is generated upon a crash

- Works for locks
- Maybe not for leader election

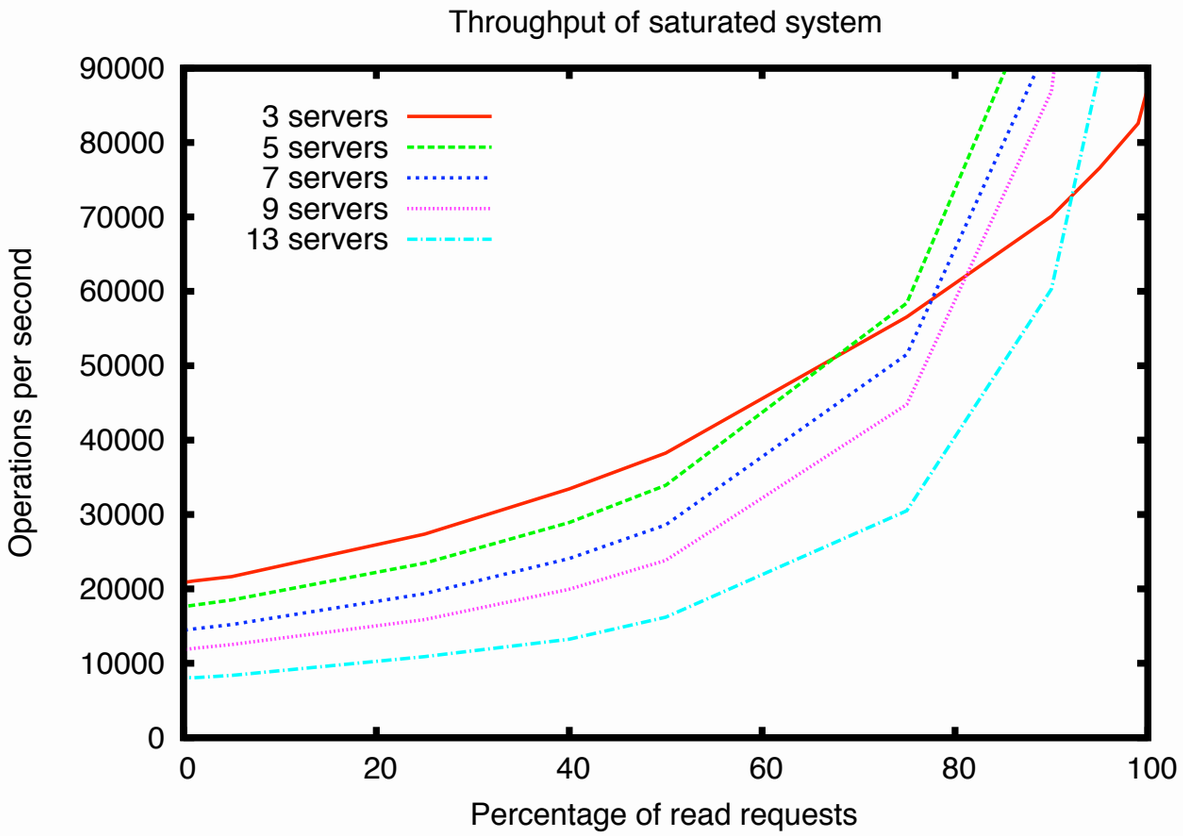
  ✓ One client is notified of a leader change

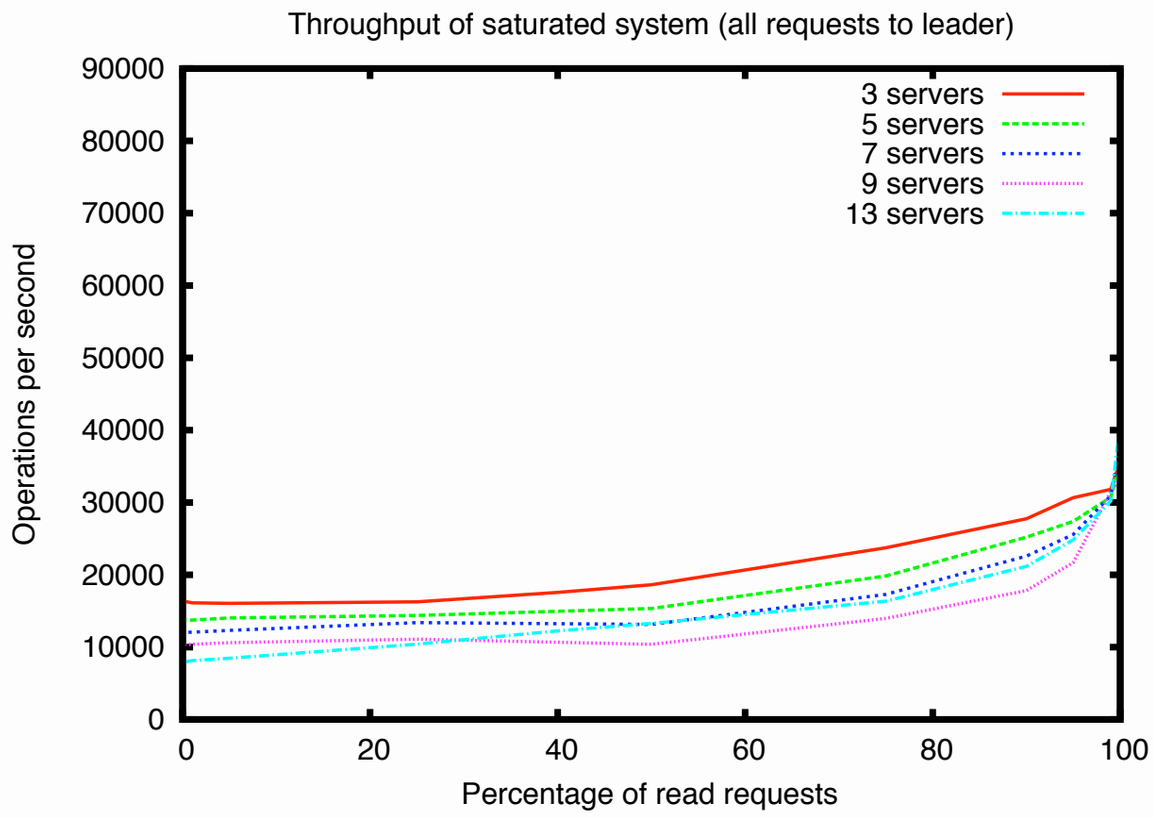
Evaluation

Evaluation

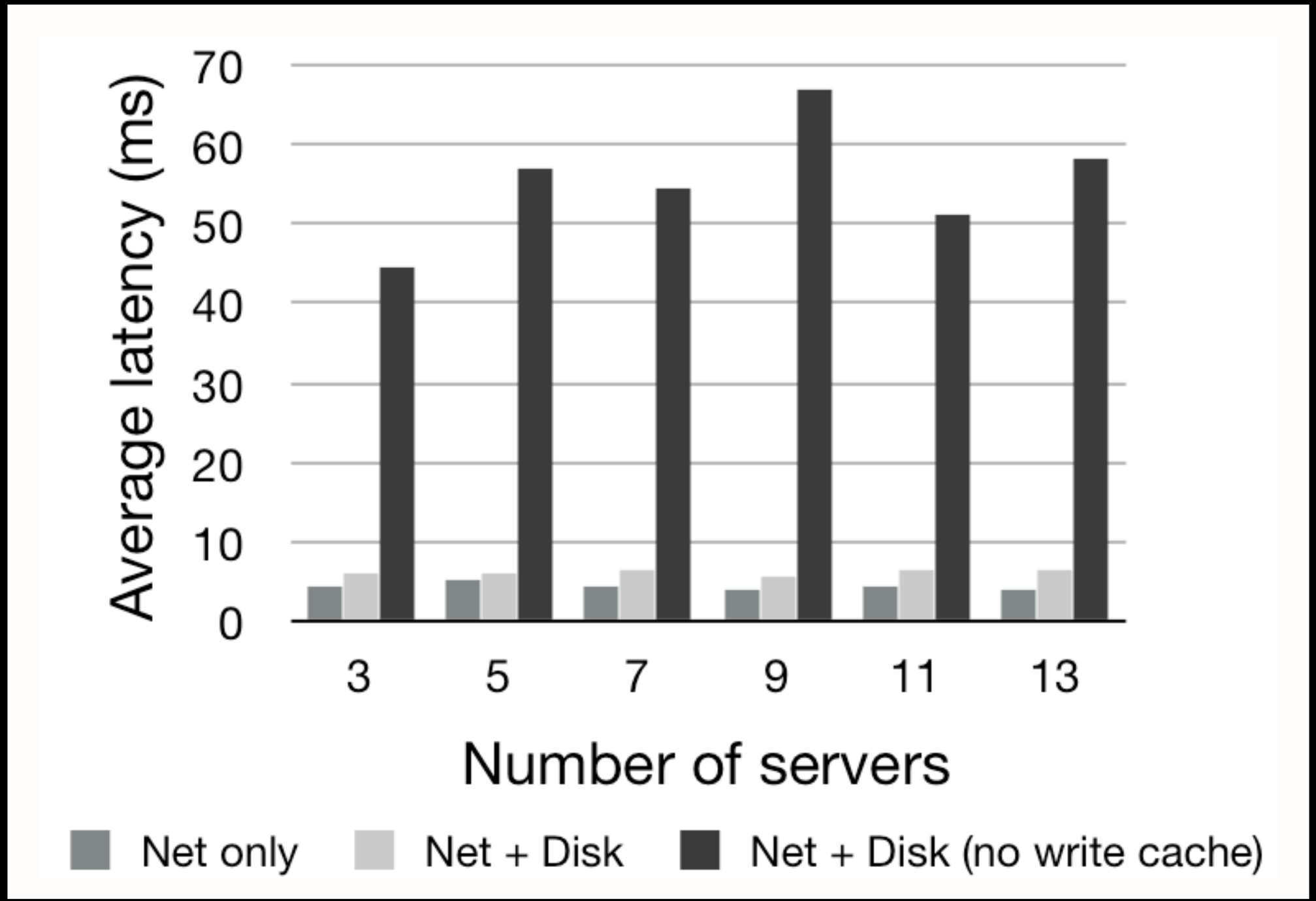
- Cluster of 50 servers
- Xeon dual-core 2.1 GHz
- 4 GB of RAM
- Two SATA disks

Throughput


Throughput of saturated system
Operations per second
Percentage of read requests
3 servers
5 servers
7 servers
9 servers
13 servers
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
0
20
40
60
80
100

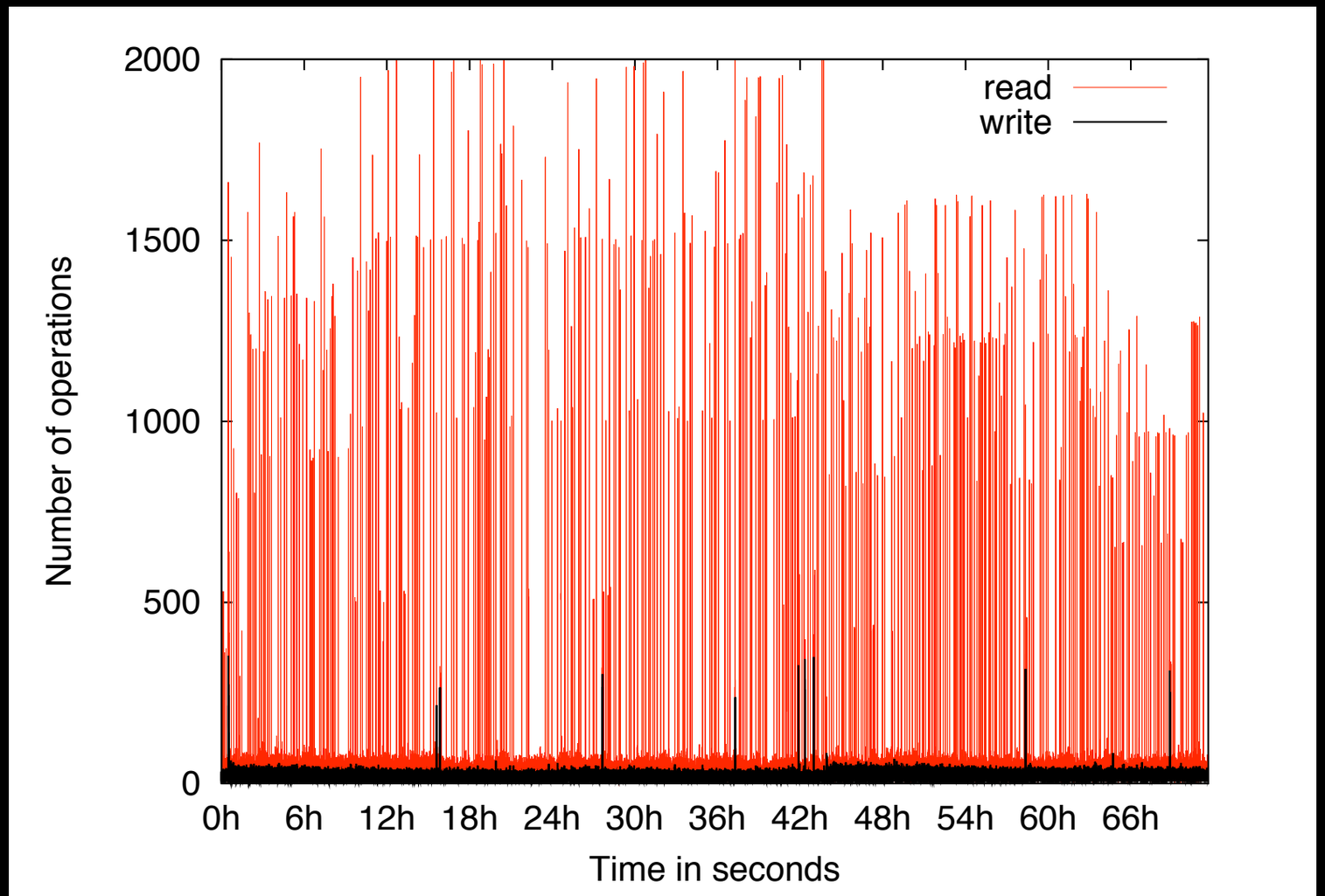

Throughput of saturated system (all requests to leader)
Operations per second
Percentage of read requests
3 servers
5 servers
7 servers
9 servers
13 servers
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
0
20
40
60
80
100

Latency


Average latency (ms)
70
60
50
40
30
20
10
0
3
5
7
9
11
13
Number of servers
Net only
Net + Disk
Net + Disk (no write cache)

Load in production

- Fetching service
- Load of a ZKserver
- Read spikes of over 2000 reads/s
- Write spikes of less less than 500 ops/s



Contributing

Steps to follow

- Watch the list for comments
- Watch issues that interest you
- Don’t be shy to chip in and ask questions
- Once you’re ready to contribute...

https://cwiki.apache.org/confluence/display/ZOOKEEPER/HowToContribute

Some important detail

- We like public communication
  - ✓ User list for questions on how to use it
  - ✓ Dev list for discussions about the code base
- Issue tracker
  - ✓ Jira https://issues.apache.org/jira/browse/ZOOKEEPER
  - ✓ Create a new jira issue if you:
    - ➡ Find a problem
    - ➡ Intend to contribute a patch

On our roadmap

- Dynamic configuration
  - ✓ Currently static
- Cross-colo deployments
  - ✓ POLE: Performance-Oriented Leader Election
  - ✓ Fault detection
- ZooKeeper as a Service
  - ✓ Multi-tenancy
  - ✓ Write scalability

Final remarks

- ZooKeeper
  - ✓ Service for coordination
  - ✓ Great experience internally and in Apache
- One of a few building blocks
  - ✓ BookKeeper
  - ✓ Hedwig
  - ✓ Omid

Questions?

http://zookeeper.apache.org