Google Developer Day2009

From Spark Plug to Drive Train: Life of an App Engine Request

Fred Sauer
June 9, 2009

Based on original presentation by Alon Levi

Google
Developer
Day2009

From Spark Plug to Drive Train: Life of an App Engine Request

This talk does not...

- ...tell you how to write an App Engine app
- ...provide any code samples
- ...include programming language specific material

What we will cover today...

- Overview of App Engine platform
- Understand components in the stack
- Explore our design motivations
- What this means for your apps

Agenda

- **How to design for Scale and Reliability**

- **App Engine: Design Motivations**

- **Life of a Request:**

1. Request for static content
2. Request for dynamic content
3. Requests accessing APIs

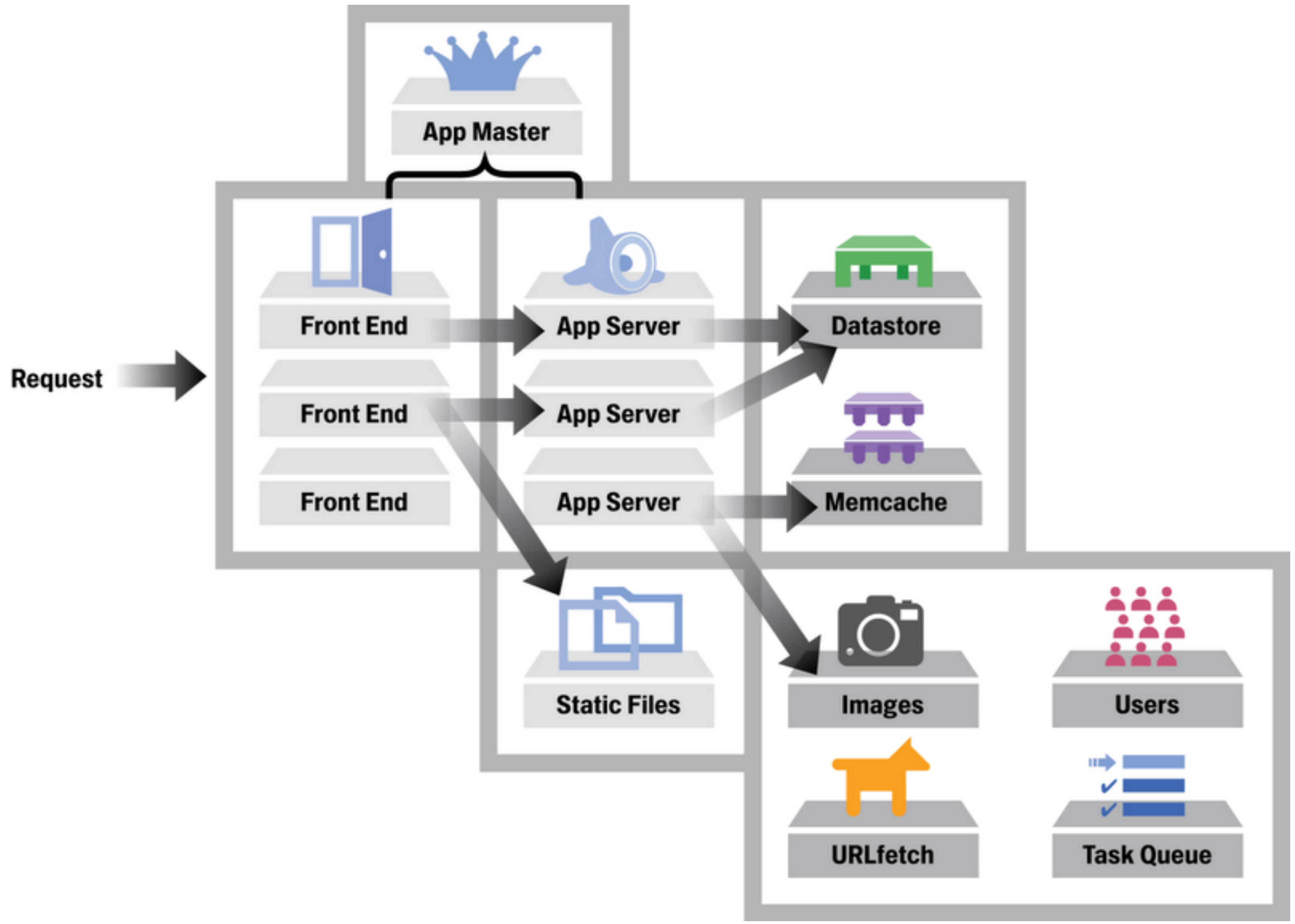
- **App Engine: Design Motivations (Recap)**

- **App Engine: The Numbers**

How to design for Scale and Reliability

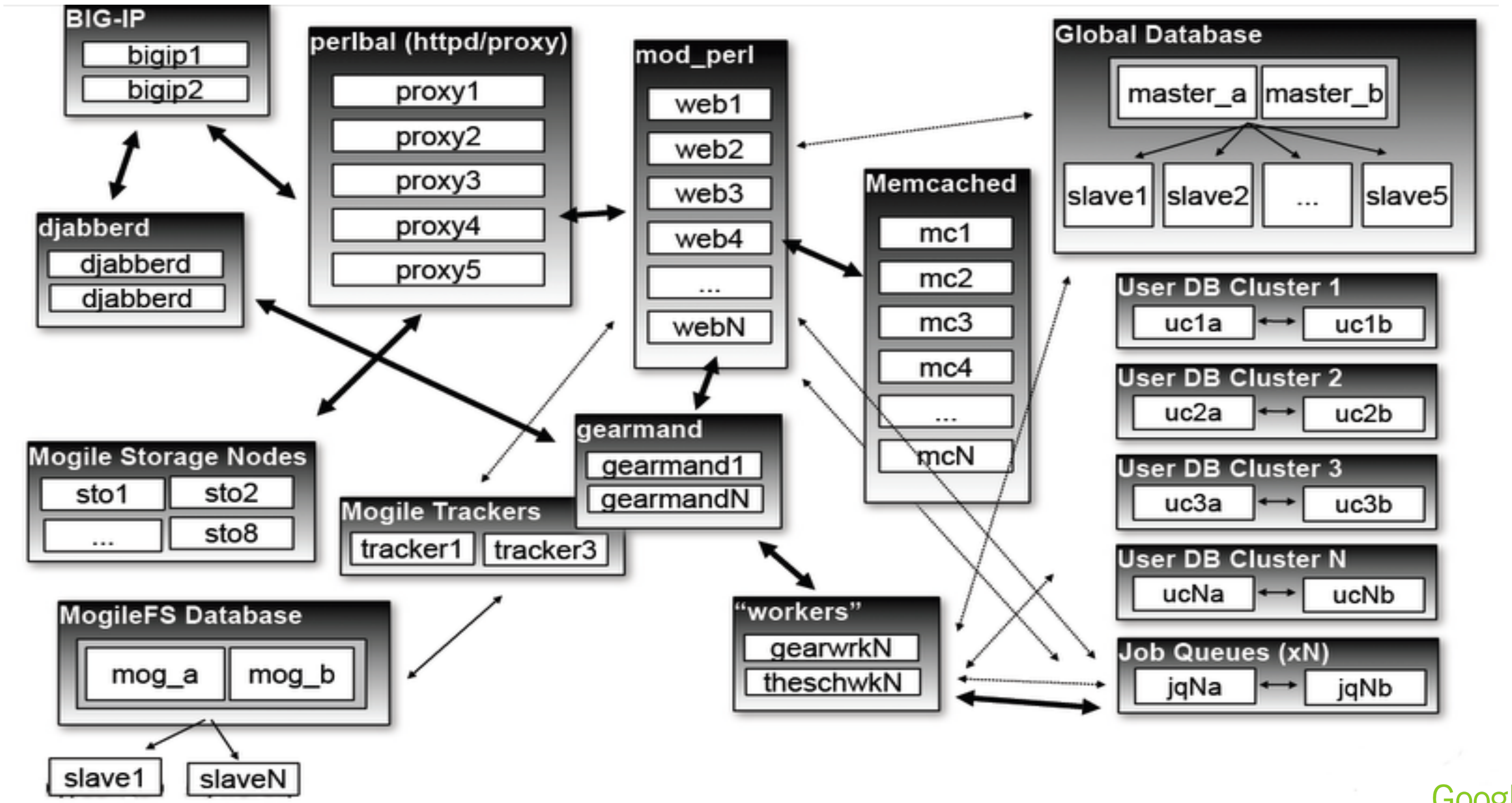
Google Developer Day2009

Google App Engine



LiveJournal circa 2007

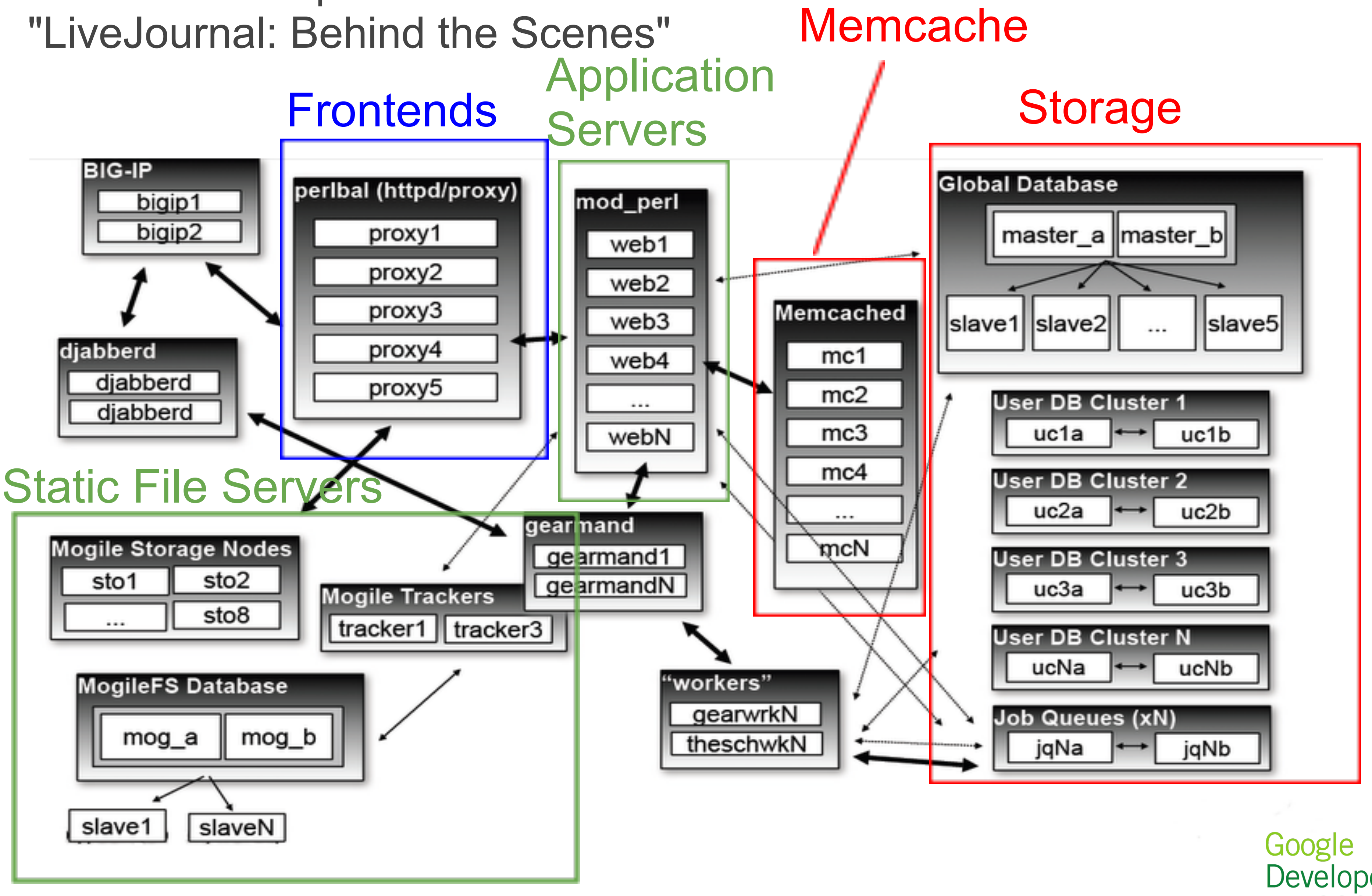
From Brad Fitzpatrick's USENIX '07 talk:
"LiveJournal: Behind the Scenes"



Google Developer Day2009

LiveJournal circa 2007

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



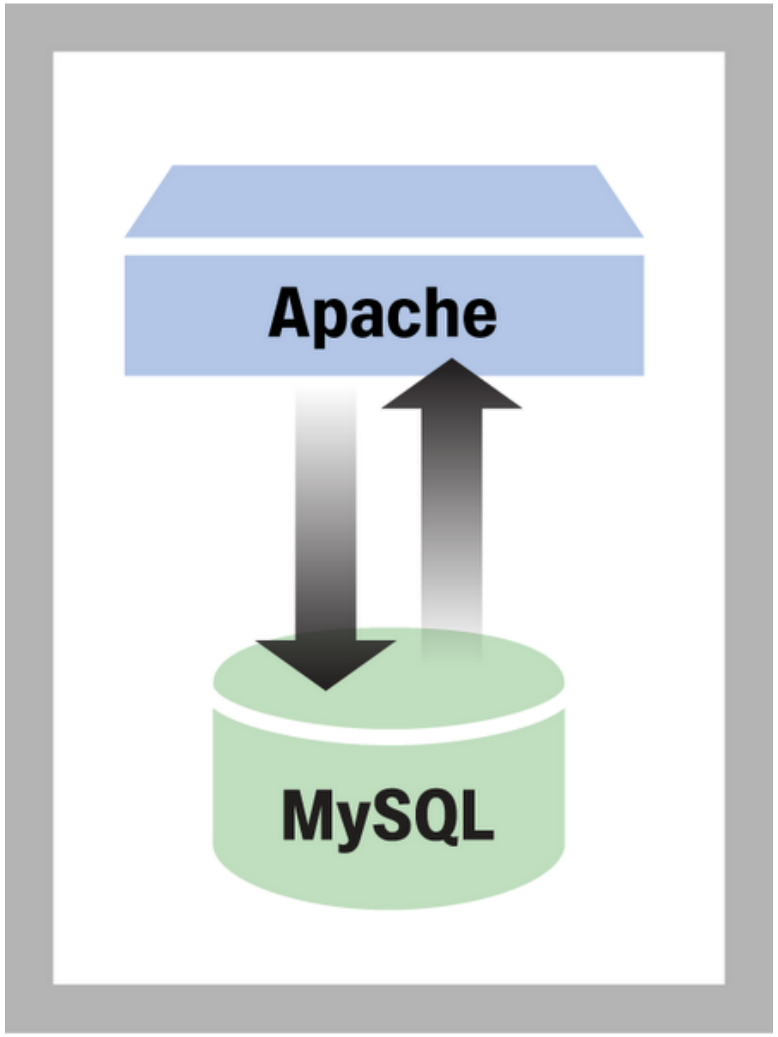
Google Developer Day2009

Basic LAMP

LAMP
- **L**inux
- **A**pache
- **M**ySQL
- **P**rogramming Language (**p**erl, **P**ython, **P**HP, ...)

Scalable?
- Shared machine for database and webserver

Reliable?
- Single point of failure (SPOF)



Dedicated Database

Database running on a separate server

Requirements:

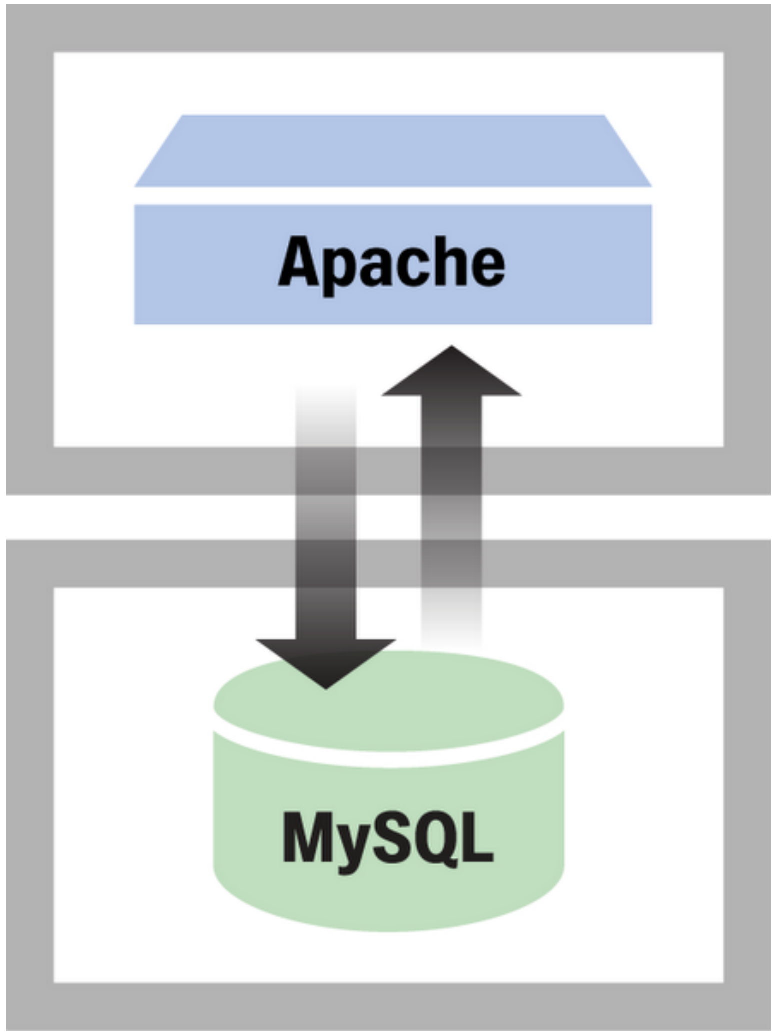
- Another machine plus additional management

Scalable?

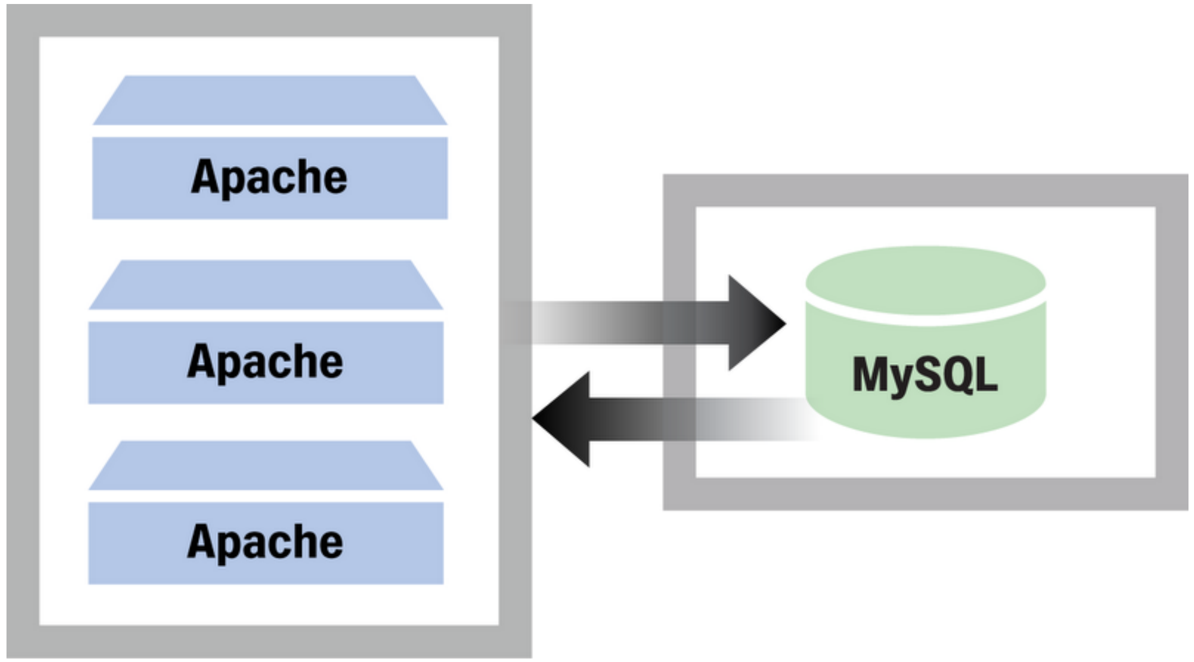
- Up to one web server

Reliable?

- **Two** single points of failure



Multiple Web Servers



Benefits:

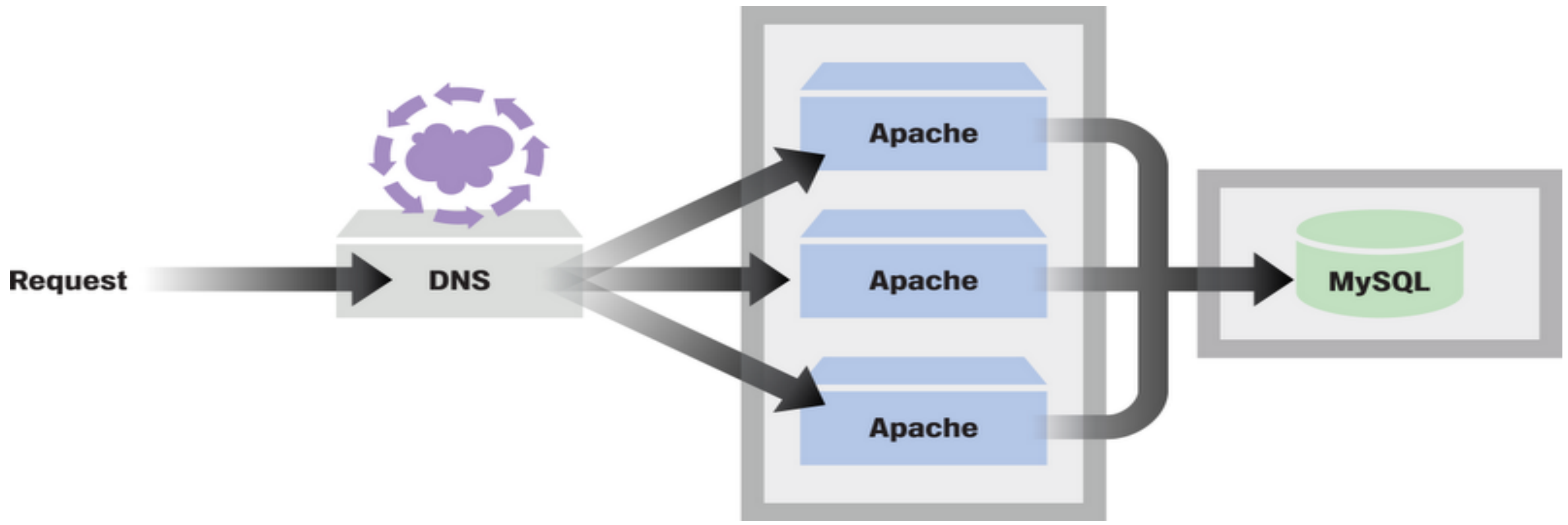
- Grow traffic beyond the capacity of one webserver

Requirements:

- More machines
- Set up load balancing

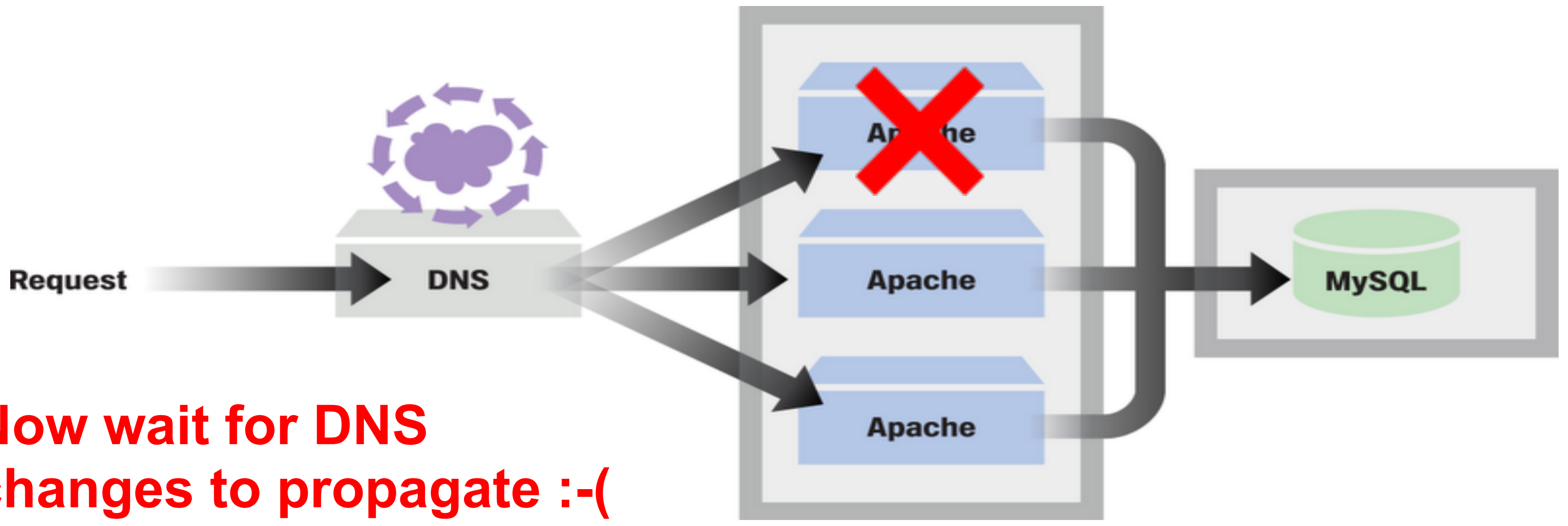
Google Developer Day2009

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

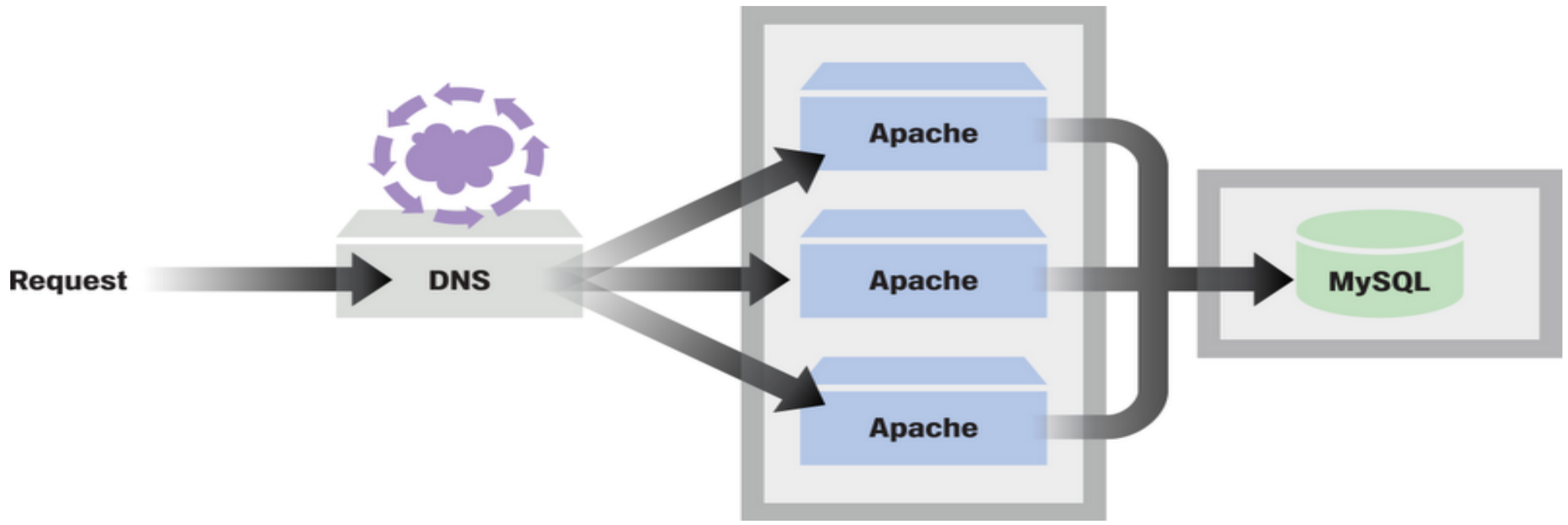
Load Balance: DNS Round Robin



Now wait for DNS changes to propagate :-(

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

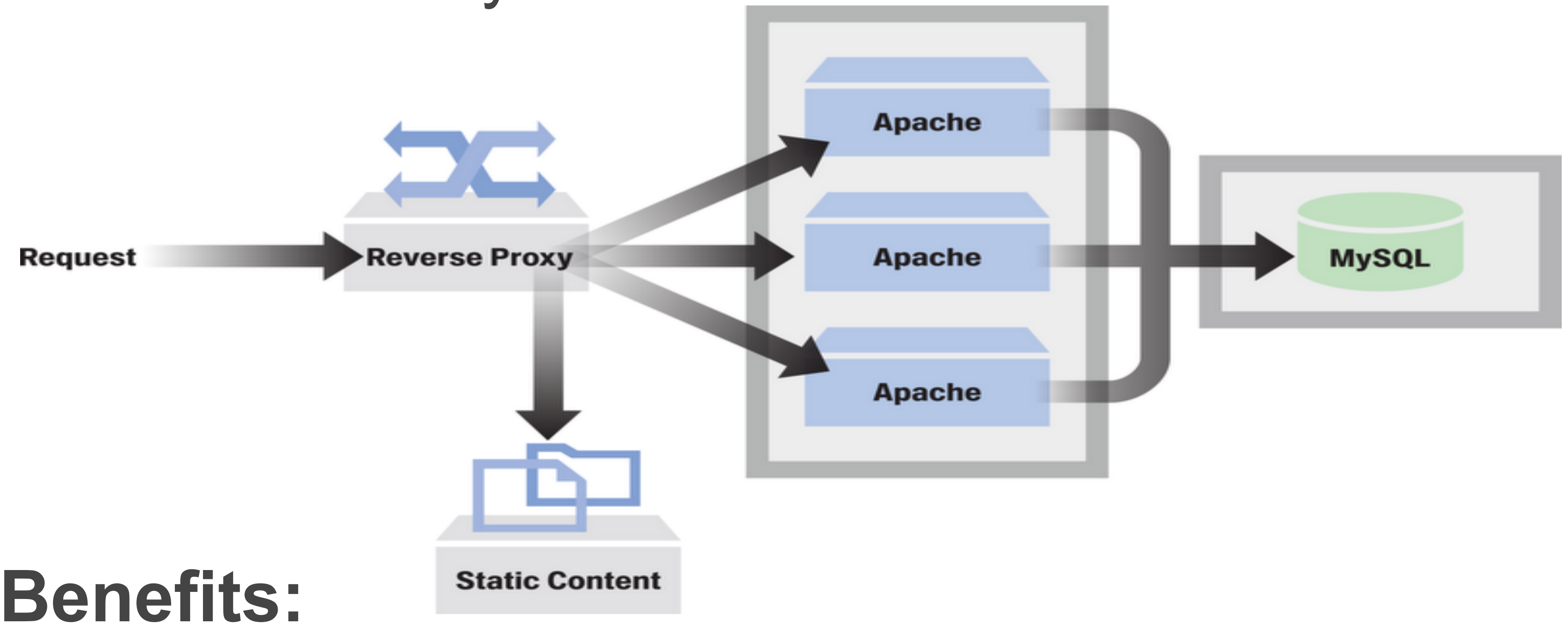
Load Balance: DNS Round Robin



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

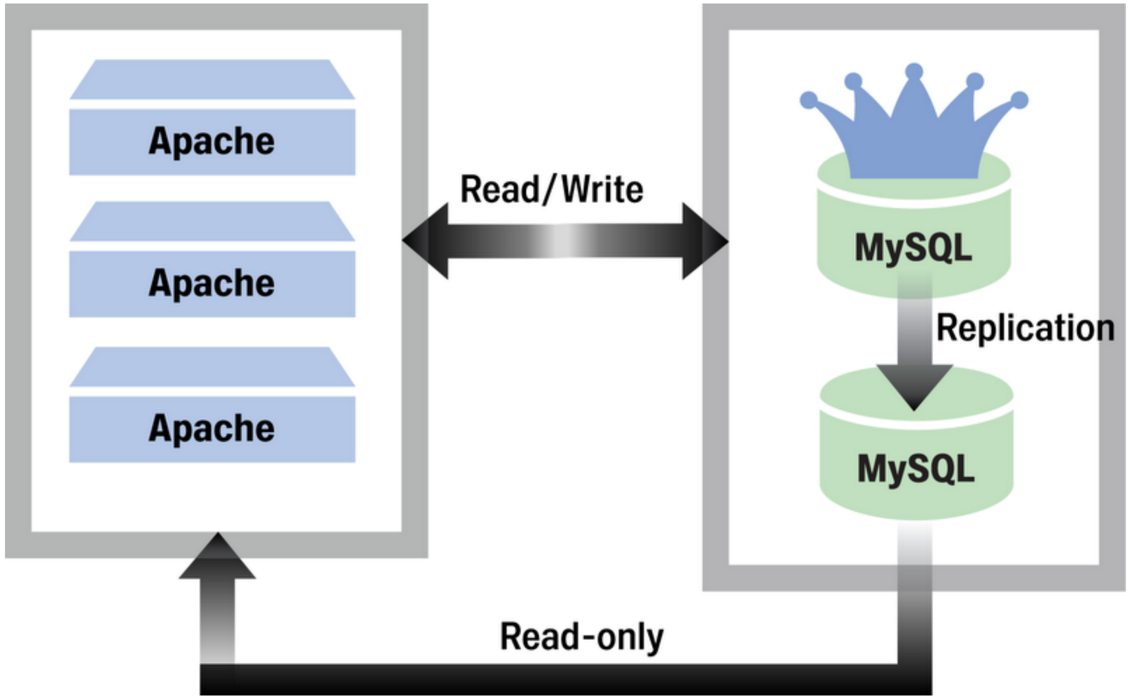
Scalable?

- 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



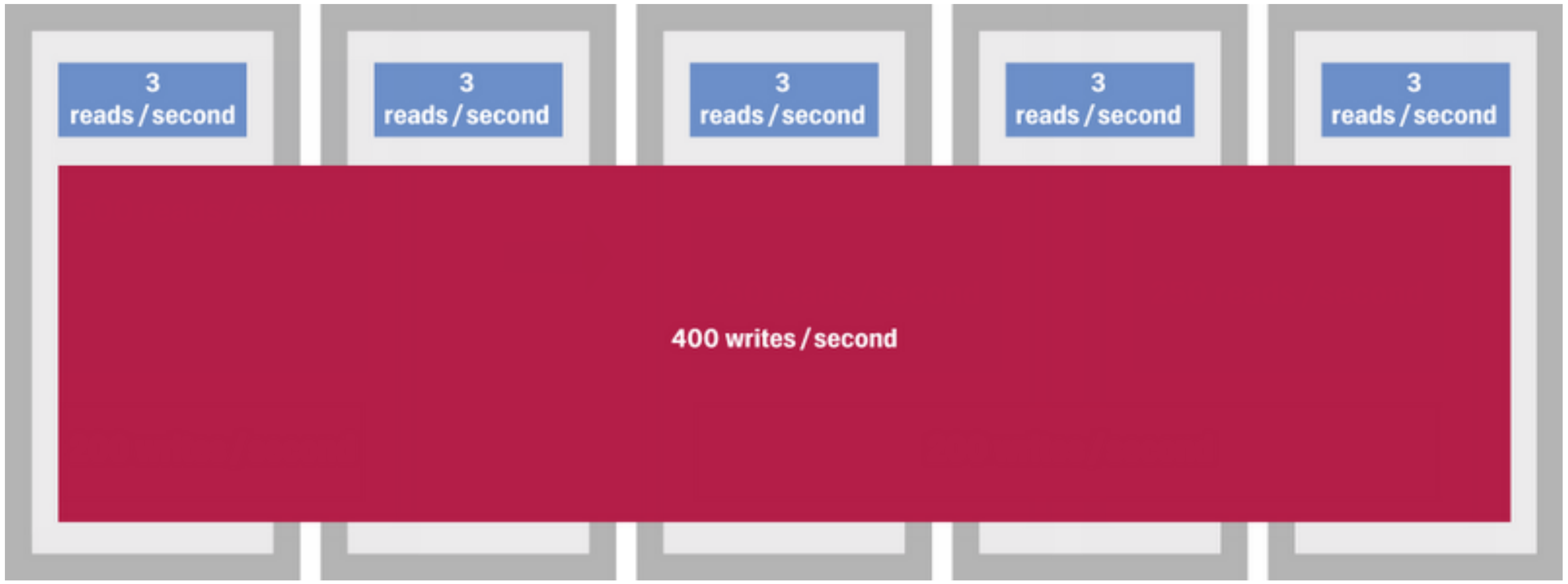
Benefits:
- Better read throughput
- Invisible to application

Requirements:
- Even more machines
- Changes to MySQL, additional maintenance

Master-Slave Database

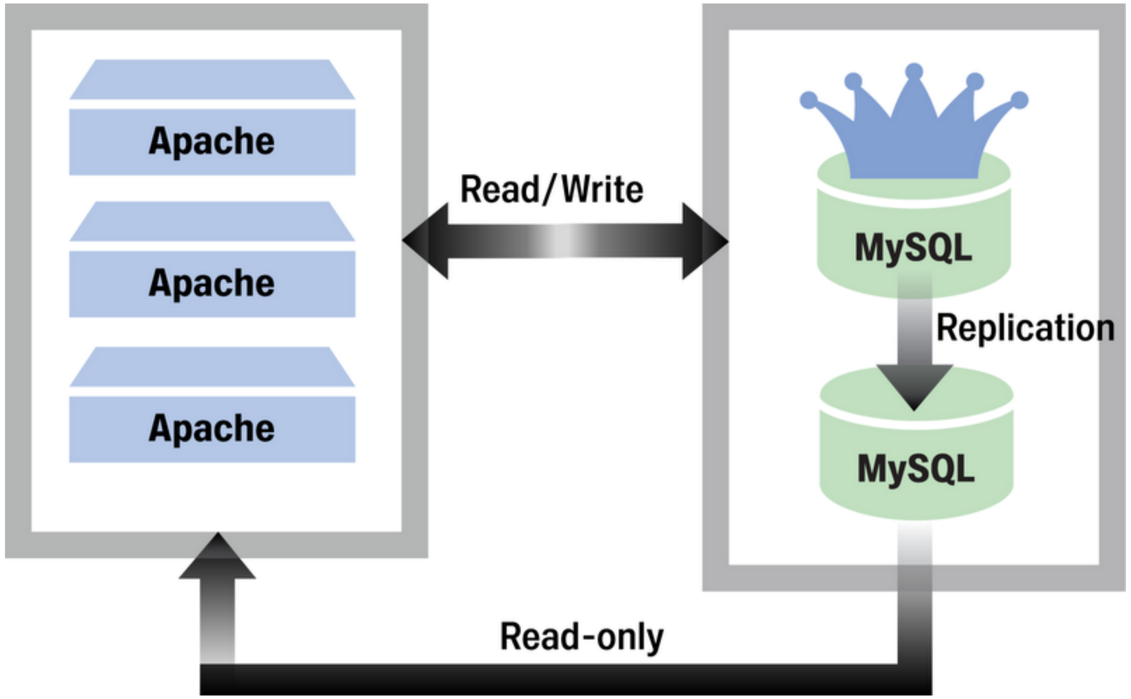
Scalable?

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



Google Developer Day2009

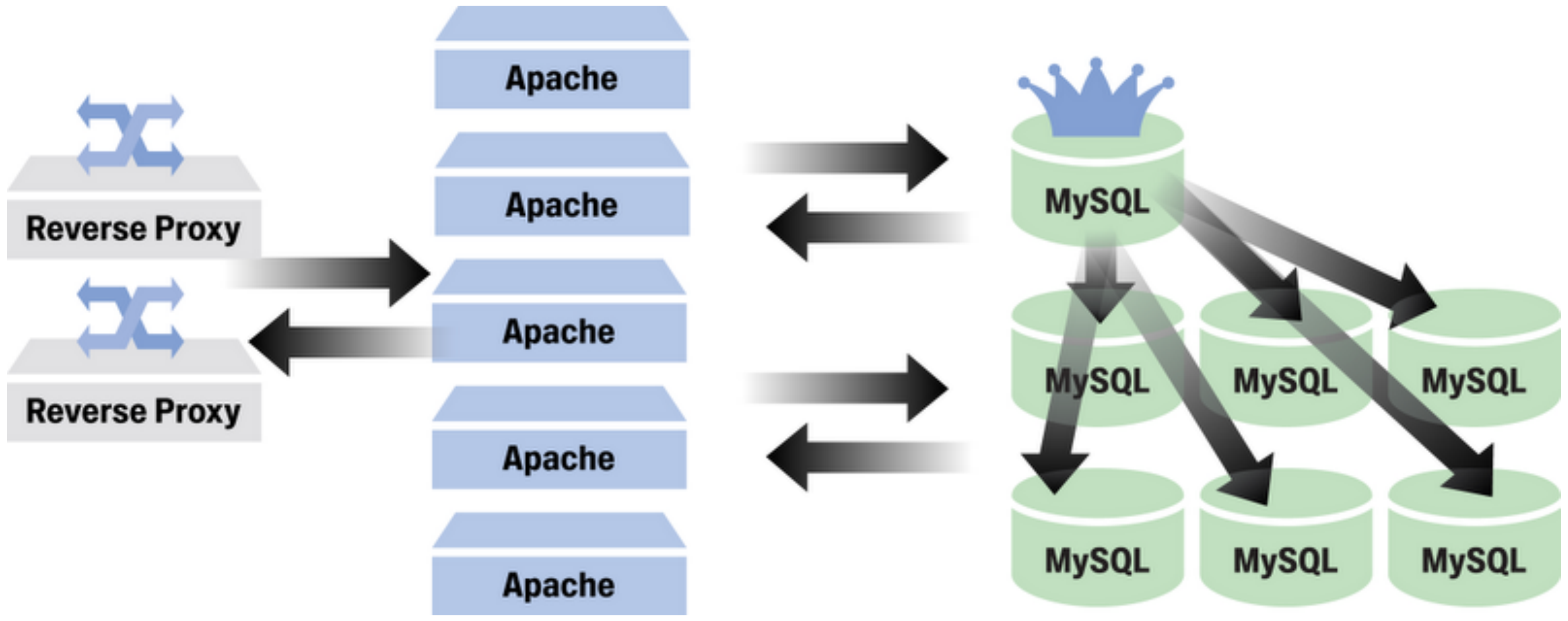
Master-Slave Database



Reliable?

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

Partitioned Database



Benefits:

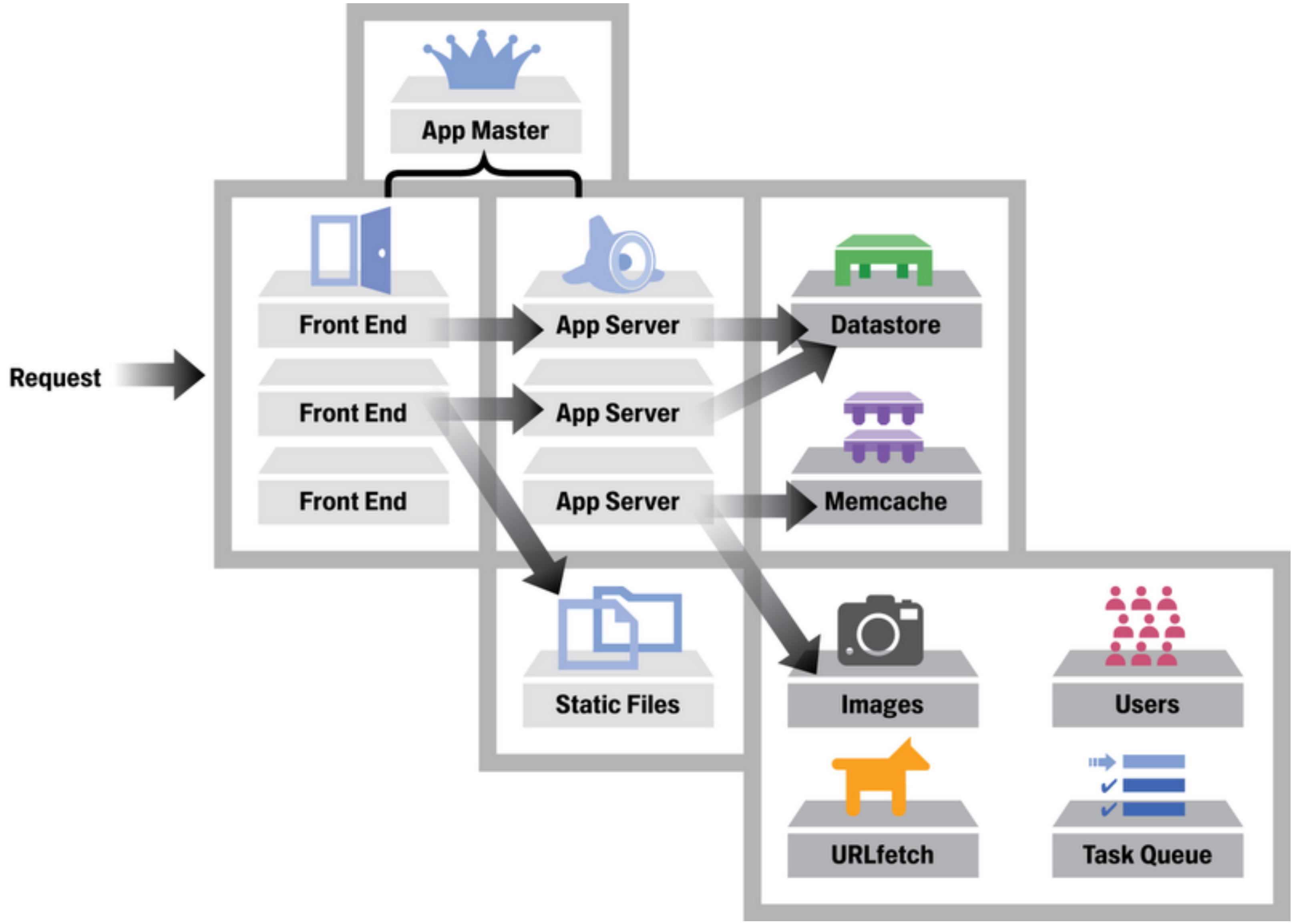
- Increase in both read and write throughput

Requirements:

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

Google Developer Day2009

Why not use Google App Engine?



App Engine:
Design Motivations

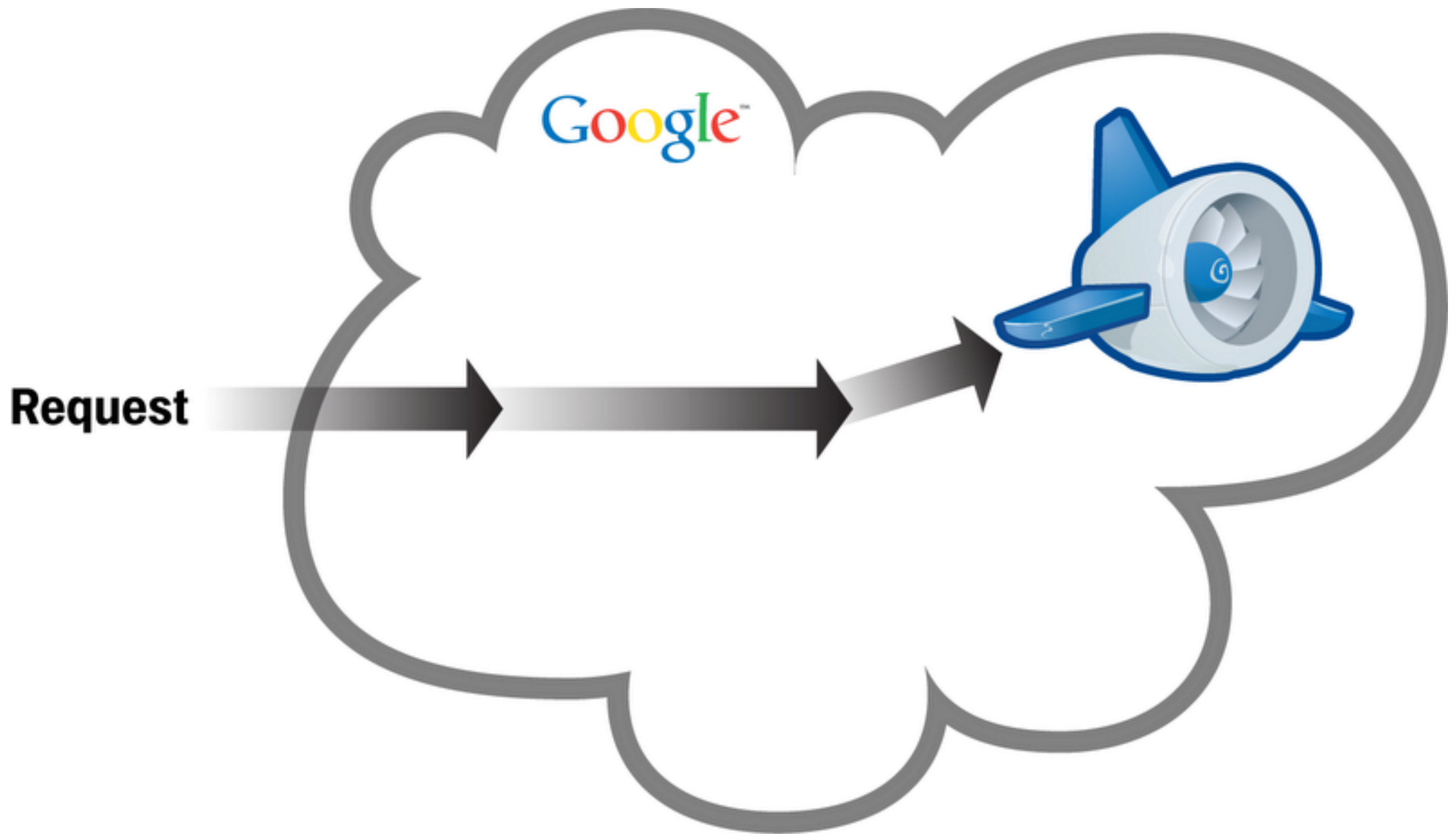
Design Motivations

- **Build on Existing Google Technology**
- **Provide an Integrated Environment**
- **Encourage Small Per-Request Footprints**
- **Encourage Fast, Efficient Requests**
- **Maintain Isolation Between Applications**
- **Encourage Statelessness and Specialization**
- **Require Partitioned Data Model**

Life of an App Engine Request

Google
Developer
Day2009

Life of an App Engine Request



- Routed to the nearest Google datacenter
- Travels over Google's network
  - Same infrastructure other Google products use
  - Lots of advantages for free

Google
Developer
Day2009

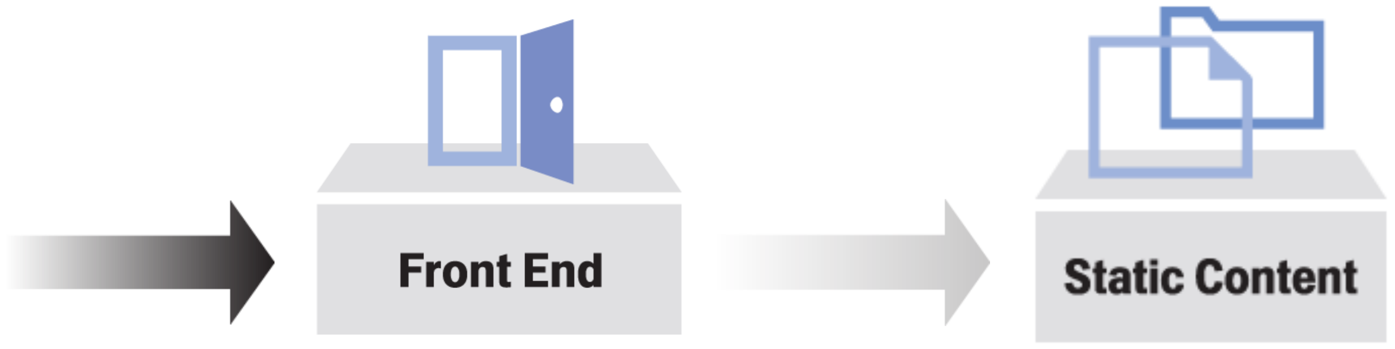
Life of an App Engine Request:

1) Request for Static Content

Google Developer Day2009

Request for Static Content

Routing at the Front End



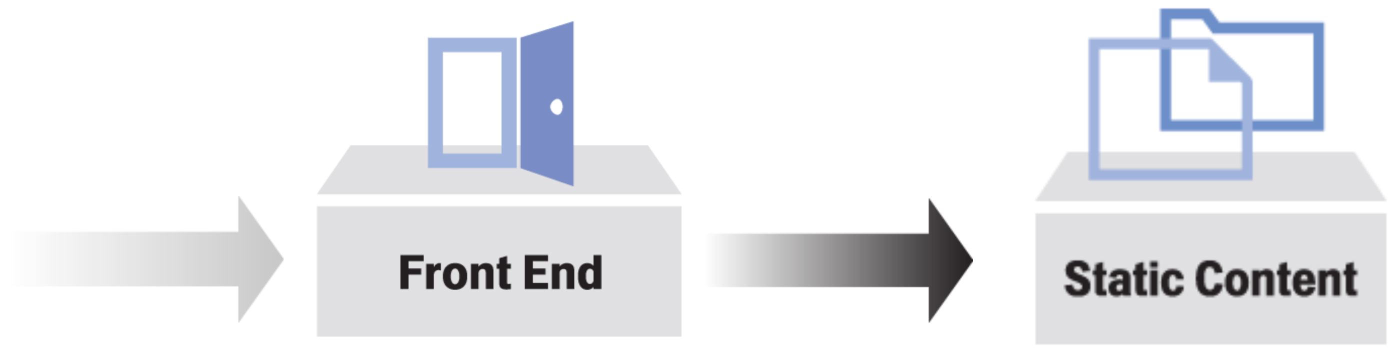
Google App Engine Front Ends

- Load balancing
- Routing

- Front Ends route static requests to specialized serving infrastructure

Request for Static Content

Static Content Servers



Google Static Content Serving

- Built on shared Google Infrastructure
- Static files are physically separate from code files

How are static files defined?

Request for Static Content

Defining static content

Java Runtime: appengine-web.xml

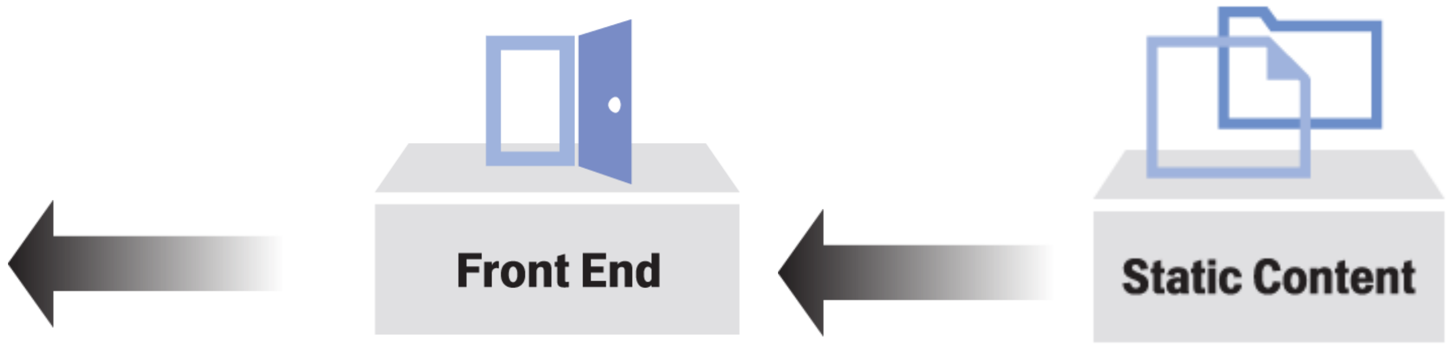
```
...
<static>
  <include path="/**.png" />
  <exclude path="/data/**.png />
</static>
...
```

Python Runtime: app.yaml

```
...
- url: /images
static_dir: static/images
OR
- url: /images/(.*)
static_files: static/images/\1
upload: static/images/(.*)
...
```

Request For Static Content

Response to the user



- Back to the Front End and out to the user
    - Front End handles connection to the user
    - Frees up Static Content server

- Specialized infrastructure
    - App Server runtimes don't serve static content

Life of an App Engine Request:

2) Request for Dynamic Content

Google Developer Day2009

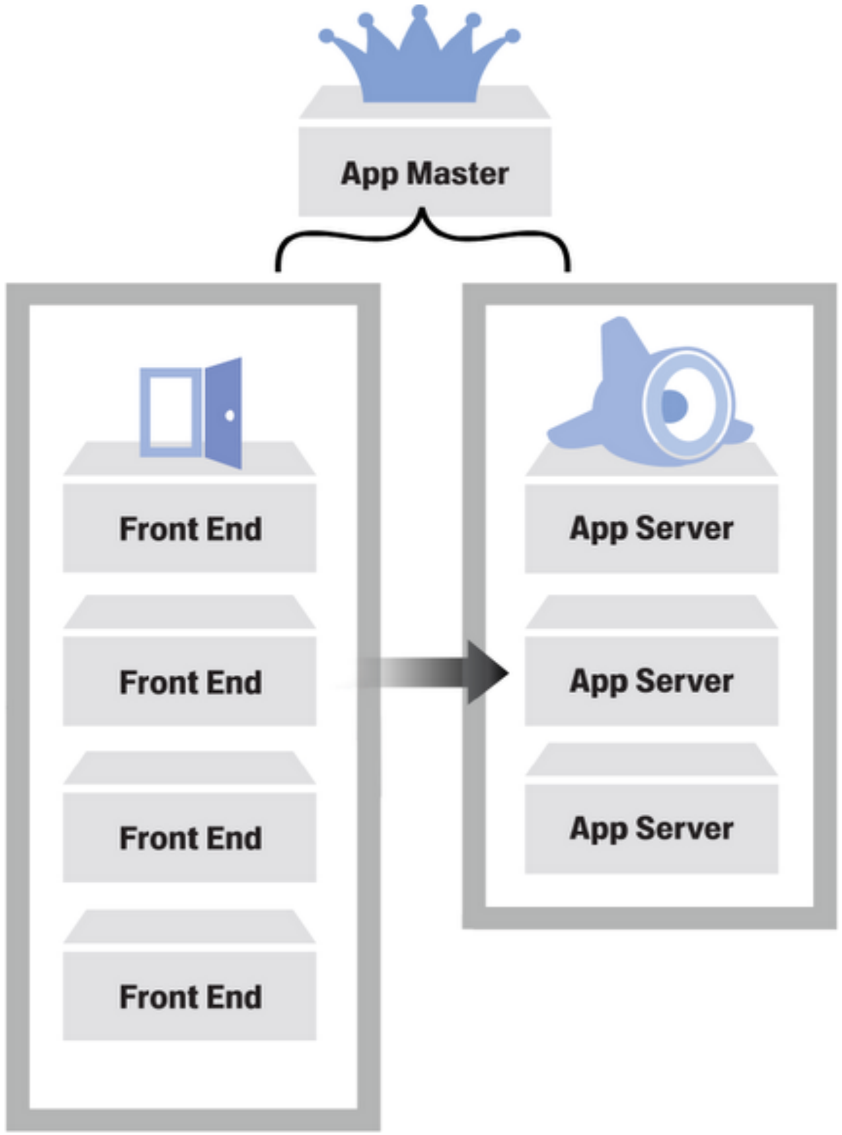
Request for Dynamic Content

Front Ends, App Servers and App Master

Front Ends
- Route dynamic requests to **App Servers**

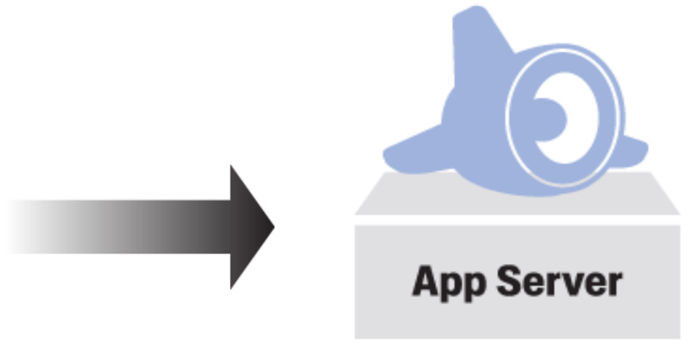
App Servers
- Serve dynamic requests
- Where your code runs

App Master
- Schedules applications
- Informs Front Ends



Request for Dynamic Content

App Server

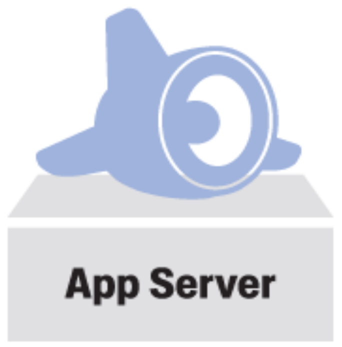


1. Checks for cached runtime
    - ○ If it exists, no initialization
2. Execute request
3. Cache the runtime

Consequences / Opportunities

- Slow first request, faster subsequent requests
- Optimistically cache data in your runtime!

App Servers - What do they do?



- Many applications
- Many concurrent requests
  - Smaller footprint + faster requests = more apps
- Enforce Isolation
  - Keeps apps safe from each other
- Enforce statelessness
  - Allows for scheduling flexibility
- Service API requests
  - Provides access to other services

Life of an App Engine Request:

3) Requests accessing APIs

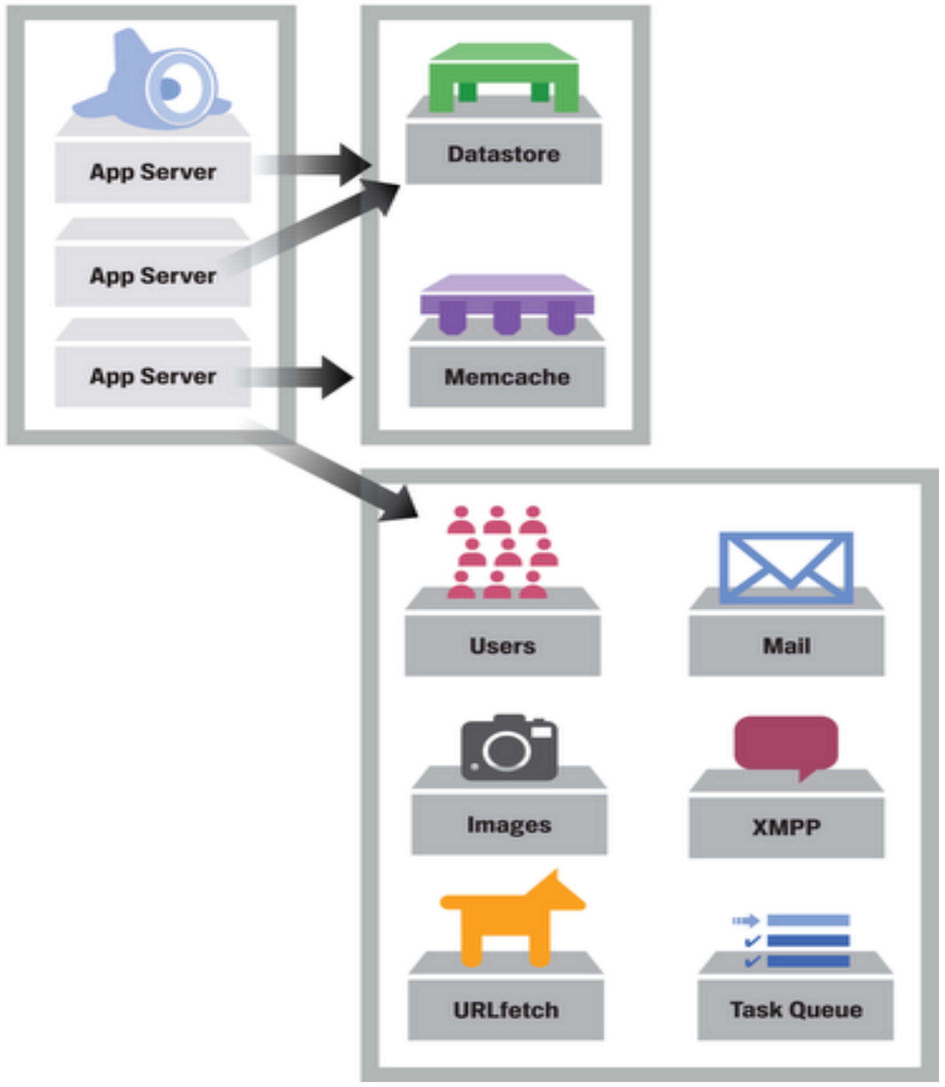
Google Developer Day2009

Requests accessing APIs

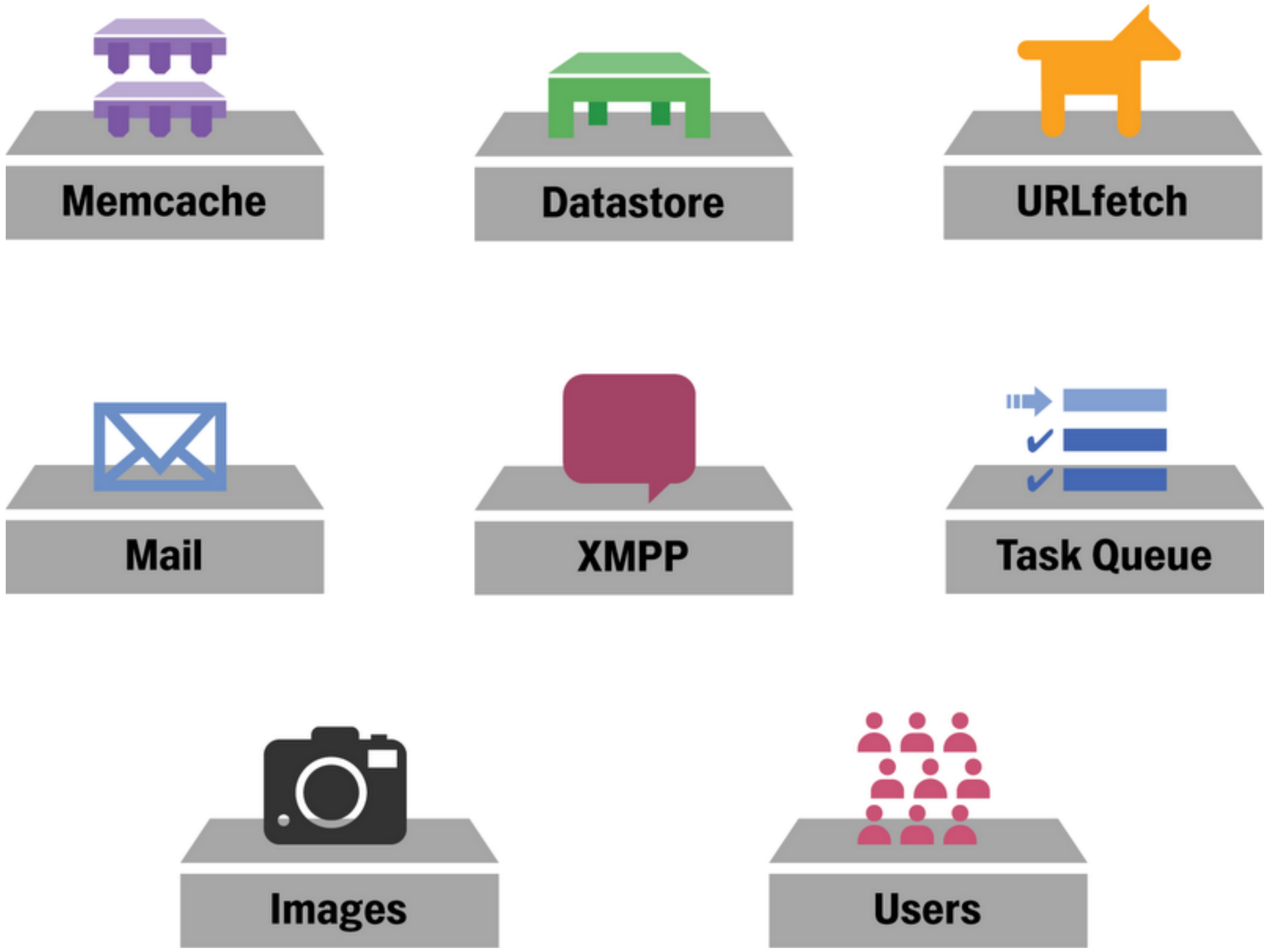
App Servers

1. App issues API call
2. App Server accepts
3. App Server blocks runtime
4. App Server issues call
5. Returns the response

- Use APIs to do things you don't want to do in your runtime, such as...

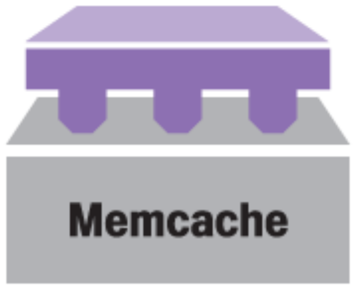


APIs


Memcache
Datastore
URLfetch
Mail
XMPP
Task Queue
Images
Users

Memcacheg

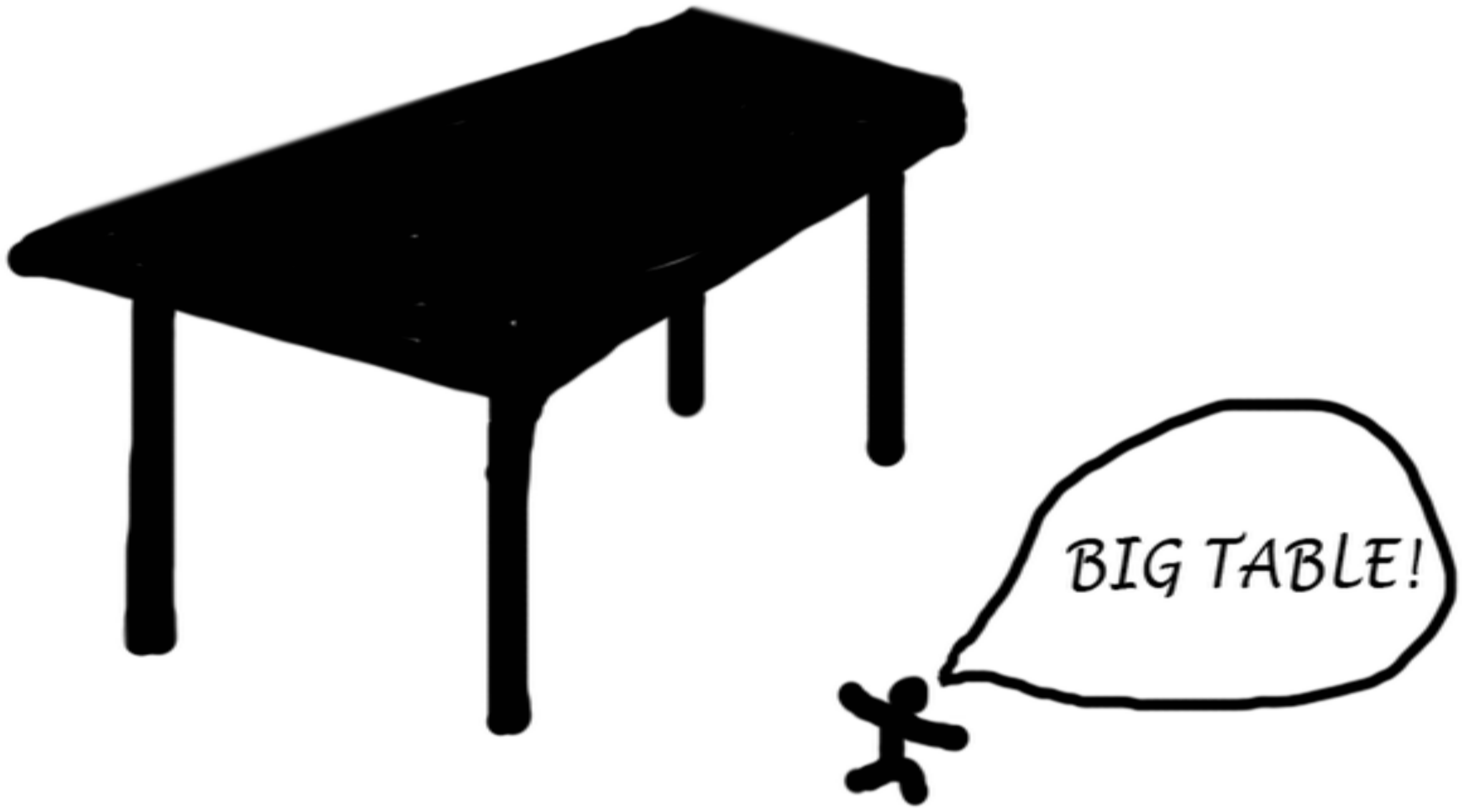
A more persistent in-memory cache



- Distributed in-memory cache
- Very fast
- memcacheg
  - Like memcached
  - Also written by Brad Fitzpatrick
  - adds: set_multi, get_multi, add_multi
- Optimistic caching
- Very stable, robust and specialized

The App Engine Datastore

Persistent storage



http://labs.google.com/papers/bigtable.html

The App Engine Datastore

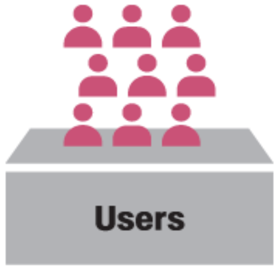
Persistent storage

- Your data is already partitioned on day one
  - Use Entity Groups
- Explicit Indexes make for fast reads
  - But slower writes
- Replicated and fault tolerant
  - On commit: ≥3 machines
  - Geographically distributed (shortly thereafter)
- Bonus: Keep globally unique IDs for free

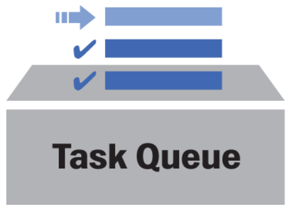
Other APIs

- GMail

- Gadget API

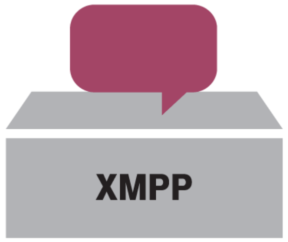


- Google Accounts



- On roadmap

- Picasaweb



- Google Talk

App Engine:
Design Motivations (Recap)

Google Developer Day2009

Build on Existing Google Technology

creative commons licensed photograph from cote

Provide an Integrated Environment

Why?

- Manage all apps together

What it means for your app:

- Encouraged to follow best practices
- Some restrictions
- Use our tools

Benefits:

- Use our tools
- Admin Console
- All of your logs in one place
- No machines to manage or count
- Easy deployment

Encourage Small Per-Request Footprints

Why?

- Better utilization of App Servers
- Fairness

What it means for your app:

- Less Memory Usage
- Limited CPU and wall clock time

Benefits:

- Better use of resources

Encourage Fast, Efficient Requests

Why?

- Better utilization of App Servers
- Fairness between applications
- Routing and scheduling agility

What it means for your app:

- Use runtime caching
- Request deadlines

Benefits:

- Optimistically share state between requests
- Better throughput
- Fault tolerance
- Better use of resources

Maintain Isolation Between Apps

Why?

- Safety
- Predictability

What it means for your app:

- Certain system calls unavailable

Benefits:

- Security
- Performance

Encourage Statelessness and Specialization

Why?

- App Server performance
- Scheduling flexibility
- Load balancing
- Fault tolerance

What this means for you app:

- Use API calls

Benefits:

- Automatic load balancing
- Fault tolerance
- Less code for you to write
- Better use of resources

Require Partitioned Data Model

Why?

- The Datastore is distributed

What this means for your app:

- Data model + Indexes
- Reads are fast, writes are slower
- Design for writes, enjoy fast reads

Benefits:

- Design your schema once
- No need to re-architect for scalability
- **More efficient use of CPU and memory**

App Engine:

The Numbers

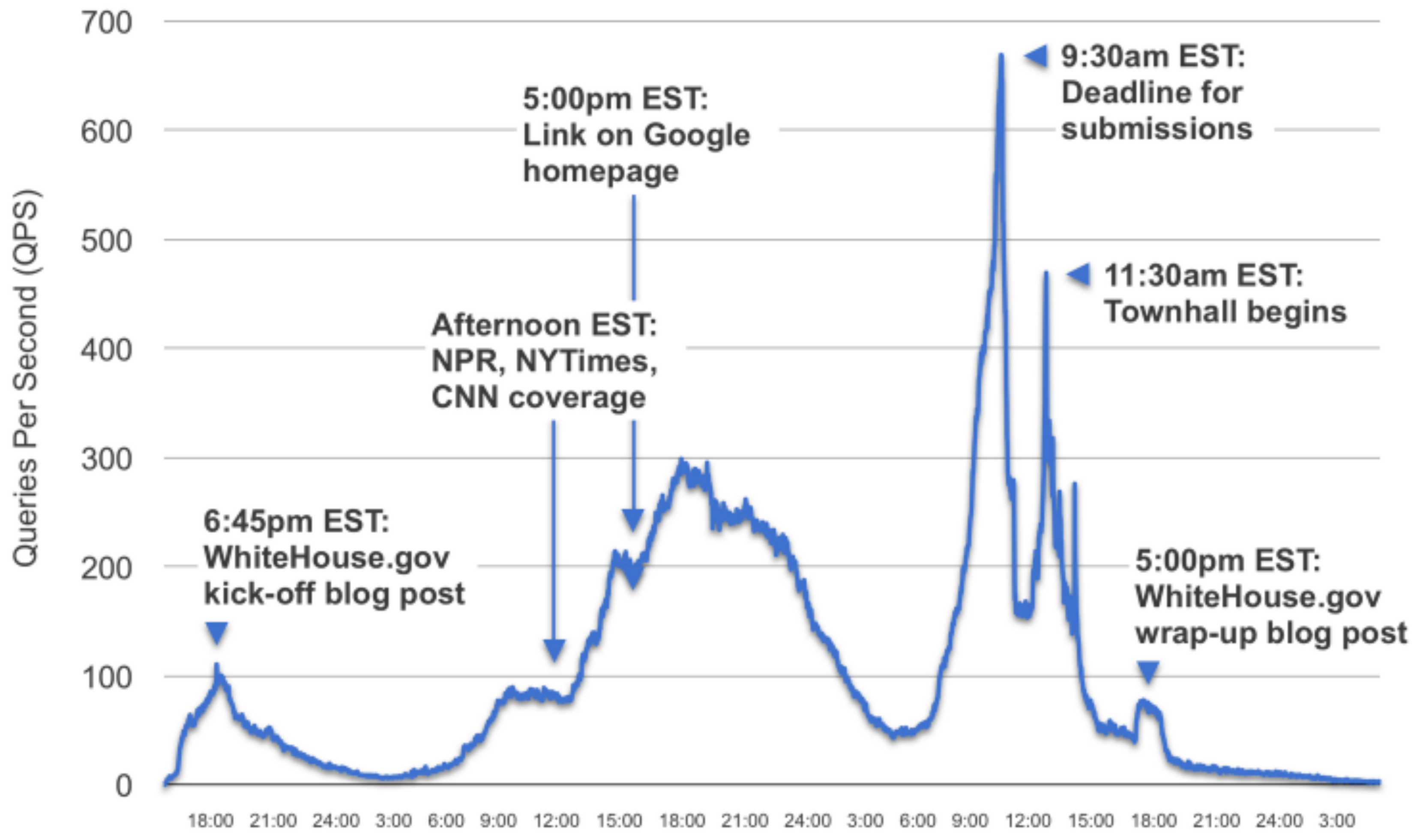
Google App Engine

- Currently, over **80K** applications
- Serving over **140M** pageviews per day
- Written by over **200K** developers
- **Two** supported languages: Python and Java
  - See also JRuby, Groovy, Scala, ...

http://groups.google.com/group/google-appengine-java/web/will-it-play-in-app-engine

Open For Questions

- The White House's "Open For Questions" application accepted **100K** questions and **3.6M** votes in under **48** hours



Thank You

Read more

http://code.google.com/appengine/

Contact info

Fred Sauer (twitter: @fredsa)Developer Advocate
fredsa@google.com

Questions

?

Google
Developer
Day2009