Advanced Networking and Distributed Systems

Introduction to Distributed Systems

GW CSCI 3907/6907
Timothy Wood and Lucas Chaufournier

Grading

Assignments = 50%

- 3 so far, 3-4 more
- Final assignment will have greater weight

Exams = 40%

- Midterm (today!)
- Quizzes (starting soon!)

Participation = 10%

- In class Q&A
- Surveys

From Networks to Distributed Systems

Networking:

- Protocols to provide efficient, reliable communication

Distributed Systems:

- ???

Why are distributed systems hard?

Challenges

Heterogeneity

- Different types of hardware/resources

Openness

- Different components need to be able to reach each other, protocols need to be understandable for others to join

Security

- can be under attack, or components could be malicious

Failure Handling

- need to recover from some components failing, can affect load distribution, single point of failure, detection of failure vs slow network

Concurrency

Challenges

Concurrency

- handle processing many tasks at once, load assignment, data consistency

Quality of Service

- performance impact of slow components

Scalability

- Should be able to make use of more resources to get better performance

Transparency

- What abstractions to make visible and what to hide

Types of Distributed Systems

Example Distributed Systems?

Req/Response

google.com

Database

Stream Processing

Twitch

Spark

Batch Processing

Hadoop

Dist Infrastructure

velocity9

AWS

blockchain

microkernel OS

Distributed System

Any multi-threaded program!

Multiple components that need to interact

Big Data Analytics

Volume: The amount of data companies want to analyze is growing tremendously

- 40 trillion gigabytes by 2020

Variety: Data is often unstructured and/or user generated

- Tweets, videos, biometrics, much more

Velocity: Analysis must be fast to be useful

- 1TB of new data generated by NY Stock exchange each day

Map Reduce & Hadoop

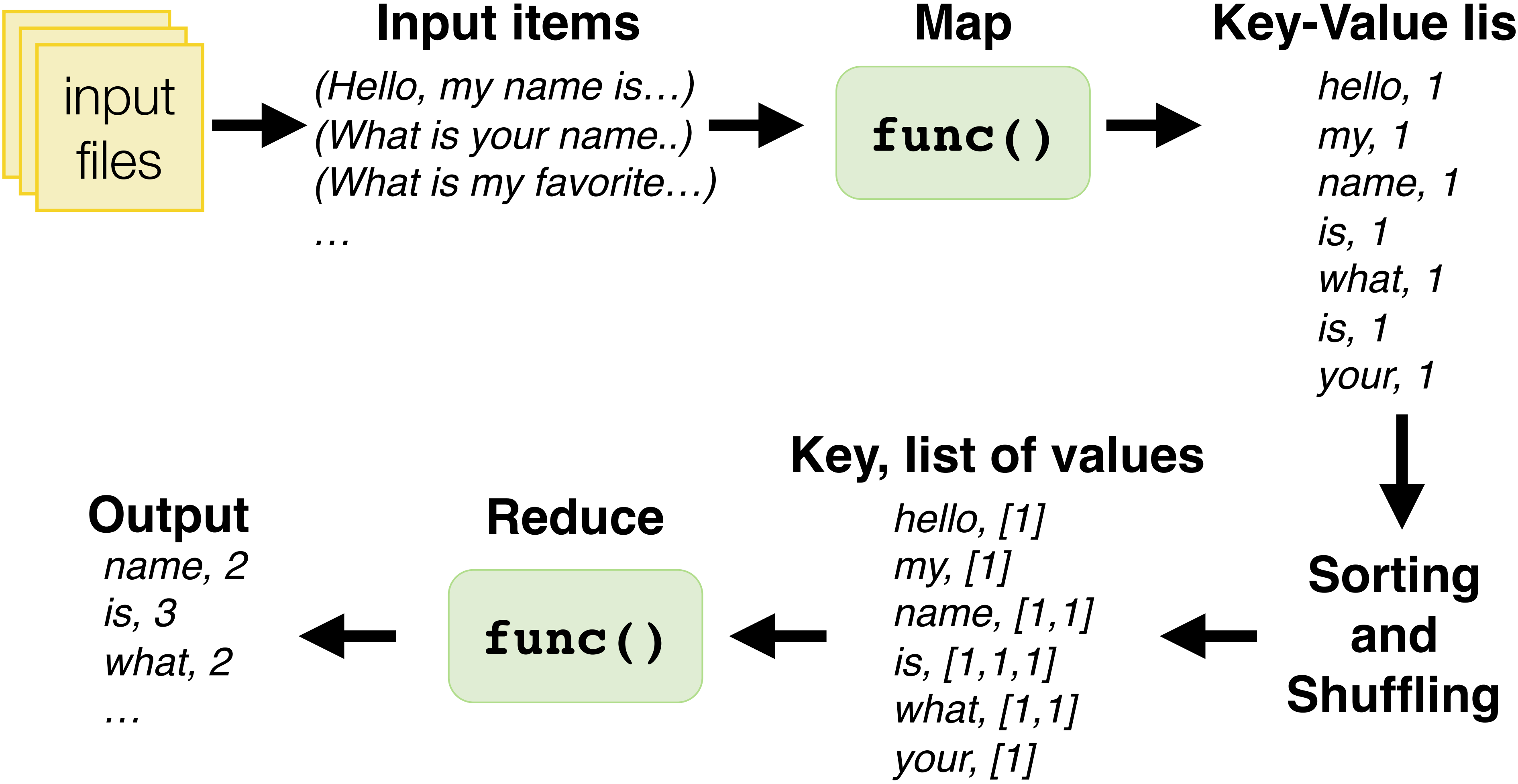
Map Reduce was developed at Google

- Large scale analytics
- Uses commodity servers
- Includes a distributed storage system
- Schedules tasks close to where data is located
- Detects and repeat failed or slow tasks
- New programming model: Map & Reduce

Hadoop is an open source version of Map Reduce

- Ideas are basically interchangeable

Map Reduce Flow



input files

Input items

(Hello, my name is…)
(What is your name..)
(What is my favorite…)
…

Map

`func()`

Key-Value list

hello, 1
my, 1
name, 1
is, 1
what, 1
is, 1
your, 1

Sorting and Shuffling

Key, list of values

hello, [1]
my, [1]
name, [1,1]
is, [1,1,1]
what, [1,1]
your, [1]

Reduce

`func()`

Output

name, 2
is, 3
what, 2
…

Stream Processing

Hadoop is for **batch** processing

- Long running jobs (minutes, hours total)

Sometimes you want **stream** processing

- Continuously arriving data with millisecond scale response

Storm is basically Hadoop for streams

- Define a graph of processing nodes
- Stream data through the graph
- Manage the workers (each executing a part of the graph)
- Detect failure, carefully buffer data in queues, etc

APACHE STORM™

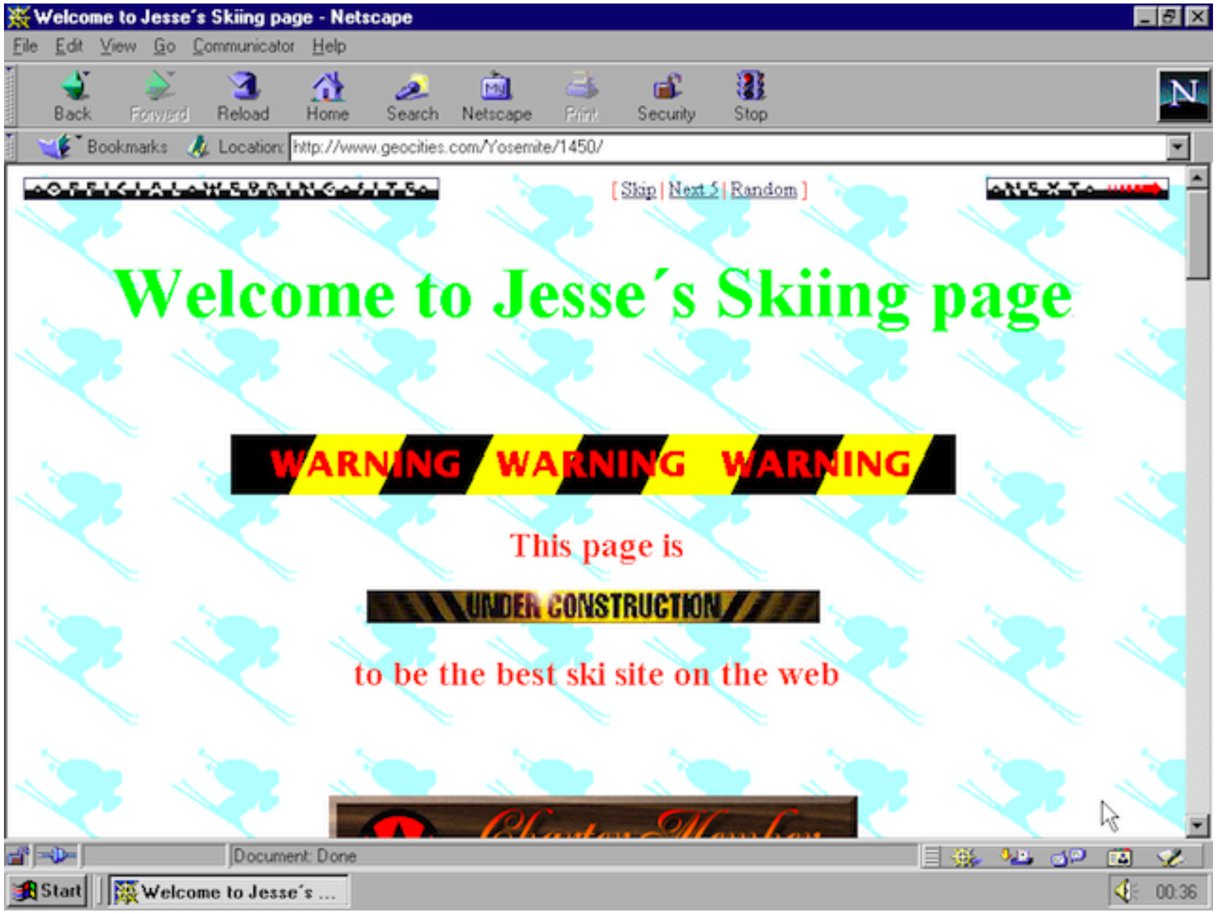
Antique Web Servers

Serve static content

- Read a file from disk and send it back to the client
- images, HTML

Dynamic Content

- CGI Bin
- executes a program
- Not very safe or convenient for development…


Welcome to Jesse's Skiing page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.geocities.com/Yosemite/1450/
OFFICIAL WEBRING SITE
[Skip | Next 5 | Random]
NEXT
Document: Done
Start
Welcome to Jesse's ...
00:36

Welcome to Jesse´s Skiing page

WARNING WARNING WARNING

This page is

UNDER CONSTRUCTION

to be the best ski site on the web

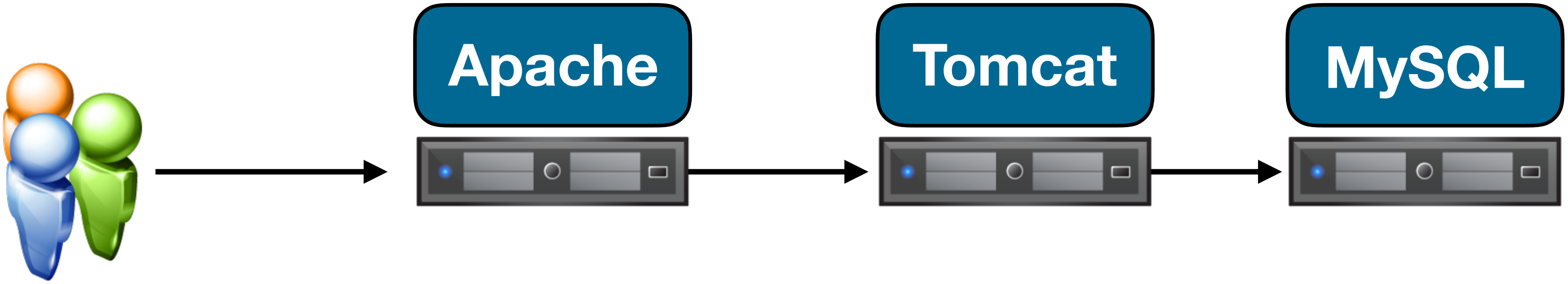
3-tier Web Applications

LAMP = Linux, Apache, MySQL, PHP

Separation of duties:

- Front-end web server for static content (Apache, lighttpd, nginx)
- Application tier for dynamic logic (PHP, Tomcat, node.js)
- Database back-end holds state (MySQL, MongoDB, Postgres)

Why divide up in this way?



Stateful vs Stateless

The multi-tier architecture is based largely around whether a tier needs to worry about state

Front-end - totally **stateless**

- There is no data that must be maintained by the server to handle subsequent requests

Application tier - maintains **per-connection state**

- There is some temporary data related to each user, e.g., my shopping cart
- May not be critical for reliability - might just store in memory

Database tier - global state

- Maintains the global data that application tier might need
- Persists state and ensures it is consistent

N-Tier Web Applications

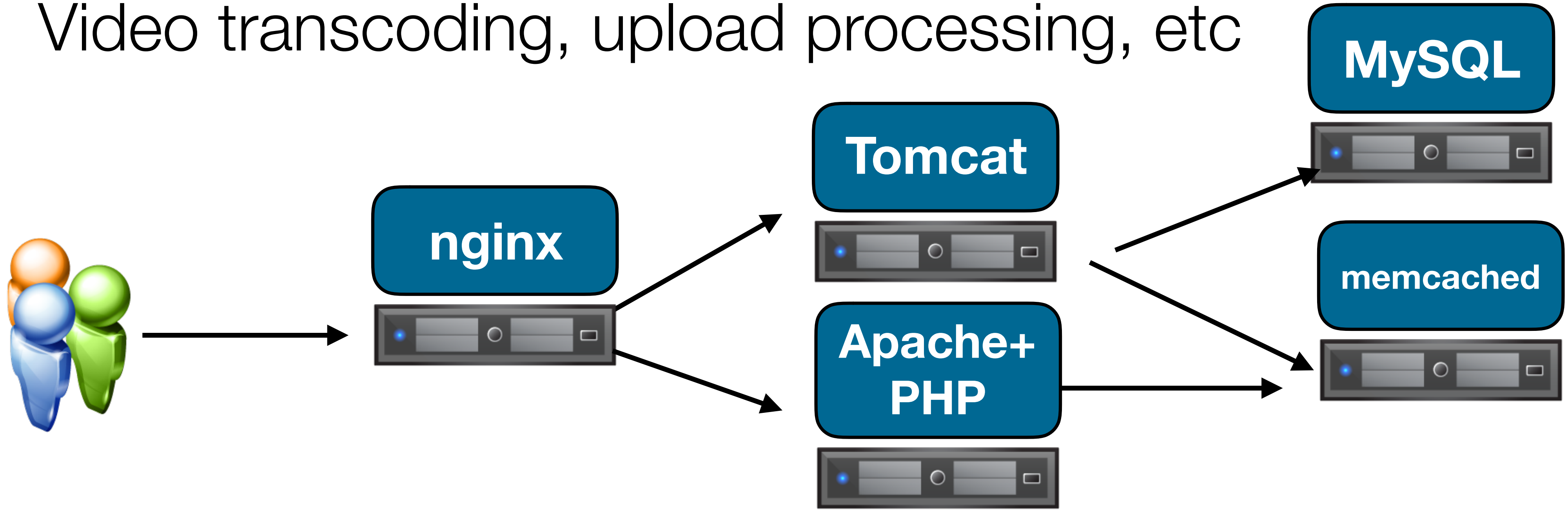
Sometimes 3 tiers isn’t quite right

Database is often a bottleneck

- Add a cache! (stateful, but not persistent)

Authentication or other security services could be another tier

Video transcoding, upload processing, etc

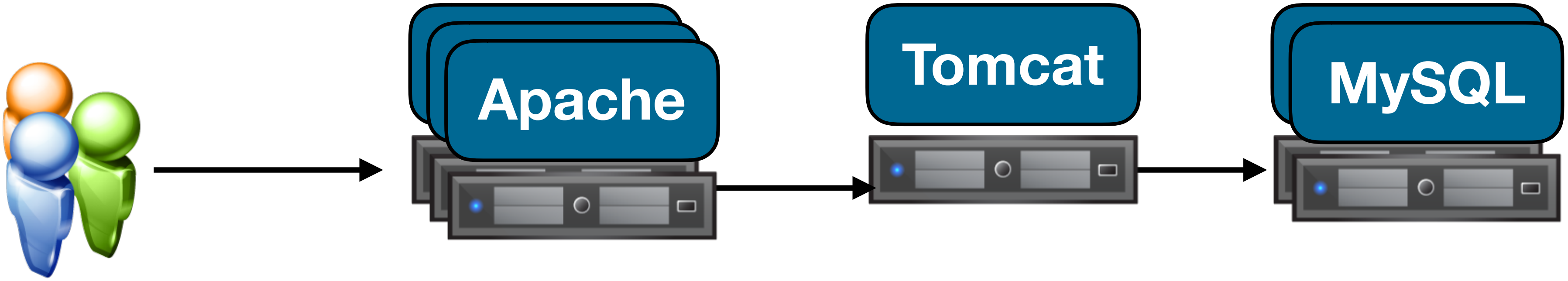


Replicated N-Tier

Replicate the portions of the system that are likely to become overloaded

How easy to scale…?

- Apache serving static content
- Tomcat Java application managing user shopping carts
- MySQL cluster storing products and completed orders



Tune number of replicas based on demand at each tier

Wikipedia: Big scale, cheap

5th busiest site in the world (according to alexa.com)

Runs on about ~ **1000** servers? (700 in 2012)

All open source software:

- PHP, MariaDB, Squid proxy, memcached, Ubuntu

Goals:

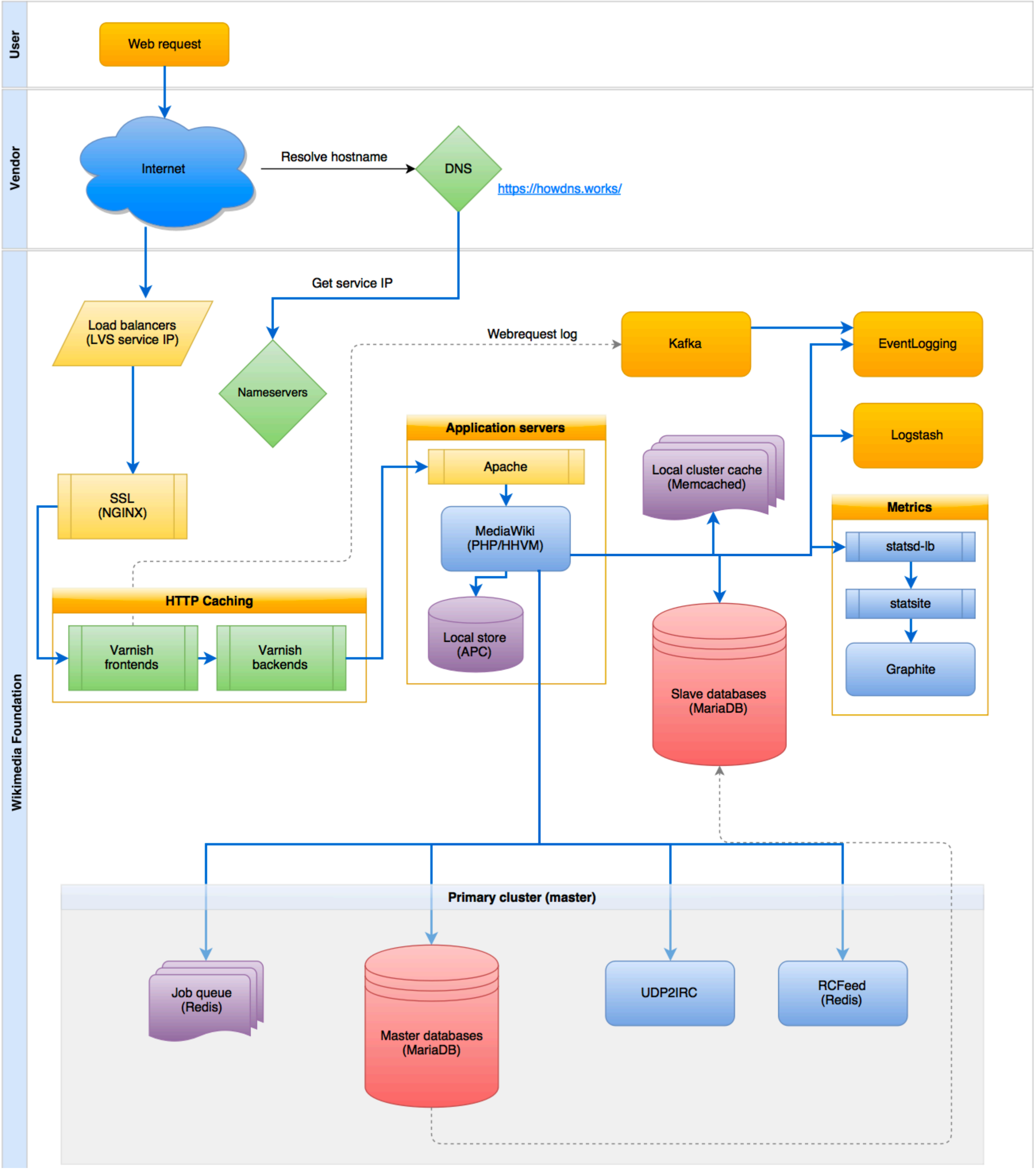
- Store lots of content (6TB of text data as of 2018)
- Make available worldwide
- Do this as cheaply as possible
- Relatively weak consistency guarantees

Stats: https://grafana.wikimedia.org

Wikipedia

Networking and application infrastructure

Revision: October 2015



User

Web request

Vendor

Internet

Resolve hostname

DNS

https://howdns.works/

Wikimedia Foundation

Get service IP

Load balancers (LVS service IP)

Nameservers

SSL (NGINX)

Webrequest log

Kafka

EventLogging

Logstash

HTTP Caching

Varnish frontends

Varnish backends

Application servers

Apache

MediaWiki (PHP/HHVM)

Local store (APC)

Local cluster cache (Memcached)

Slave databases (MariaDB)

Metrics

statsd-lb

statsite

Graphite

Primary cluster (master)

Job queue (Redis)

Master databases (MariaDB)

UDP2IRC

RCFeed (Redis)