Building & Maintaining the Building Blocks of the Web
Who am I and why should you listen to me?
What should you get out of this talk?
Building & Maintaining the Building Blocks of the Web

- Overview of Systems
- History of Hardware
- Virtual Machines & Containers in the Cloud
- Monitoring Systems
- Keeping Systems Running
Definitions

- **Machine** - Hardware that runs servers
- **Server** - Software that does something and runs on machines
- **Job** - A piece of software with a common purpose
- **Task** - An instance of a job
- **Service** - A collection of APIs
- **Request** - A call into a task to ask it to do something
- **QPS** - Queries per second
Datacenter Definitions

- A **rack** is made up of 10s of **machines**
- **Racks** stand in a **row**
- One or more **rows** form a **cluster**
- **Datacenter buildings** host multiple **clusters**
- **Datacenter campus** hosts multiple **buildings**
Overview of Systems
Most code you write looks like this:

```c
int main()
{
    doSomething();
    doSomethingElse();
    return 0;
}
```
Most code that runs today looks like this:

```c
int main() {
    while(1) {
        request = getRequest();
        result = processRequest(request);
    }
    return 0;
}
```
Services

- Services are the building blocks of modern software
- Other jobs are called into like a local library to perform a task
- Services can be big or small (micro)
- Many technologies exist to build upon
  - gRPC & Protocol Buffers
  - JSON & REST
Those technologies solve common problems:

- Serialization & Deserialization of requests & results
- Sanitization of inputs
- Definition of the API
- Error handling
- Retry handling
- Rate limiting & Throttling
- Permission ACLs (Access Control List)
A modern application might look like this:
History of Hardware
In the beginning...
Even for Google

The Original Google Storage

In 1996 Larry Page and Sergey Brin, then PhD students in Stanford EECS, working on the Digital Library Project, needed a large amount of disk space to test their PageRank™ algorithm on actual word-wide-web data. At that time 4 Gigabyte-hard disks were the largest available, so they assembled 10 of these drives into a low-cost cabinet.

In Nov 1998, Google Inc. by then operating one of the primary search engines on the web, provided replacement storage capacity to the Digital Library project so that we could move the original storage assembly into our history displays.

As of September 2000, Google, now located in Mountain View, operated 5000 PCs for searching and web crawling, using the Linux operating system.
Enter the Datacenter.....
Also for Google...
The Cloud
Lots of machines....
Also other stuff...
Modern day datacenters

- Basically a cost center
- Trade $$$ for computing power & storage
- The more efficient the better
  - This is true for both computing resources and power consumption
- Rely on regular hardware that will fail
- Sometimes run themselves
Virtual Machines & Containers in the Cloud
<table>
<thead>
<tr>
<th>Hardware Machine</th>
<th>Virtual Machine or Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>You get all the resources</td>
<td>You get some of the resources</td>
</tr>
<tr>
<td>Hard to scale</td>
<td>Easy to scale</td>
</tr>
<tr>
<td>Static configuration</td>
<td>Dynamic configuration</td>
</tr>
<tr>
<td>Slow duplication</td>
<td>Fast duplication</td>
</tr>
<tr>
<td>Don’t have to worry about other jobs running</td>
<td>Don’t have to worry about other jobs running</td>
</tr>
<tr>
<td>Less overhead</td>
<td>More overhead</td>
</tr>
<tr>
<td>Less security risk</td>
<td>Potential security risks</td>
</tr>
<tr>
<td>More performance</td>
<td>Less performance</td>
</tr>
<tr>
<td>Less efficient</td>
<td>More efficient</td>
</tr>
</tbody>
</table>
Virtual Machines vs Containers

Containers are isolated, but share OS and, where appropriate, bins/libraries...
result is significantly faster deployment, much less overhead, easier migration, faster restart
Containers

- Generally considered the better option
- But VM are still popular
- Docker & Kubernetes
- Run on all major cloud platforms
- Easy to set up
- Easy to incorporate other services
- Seen as being relatively new
But they aren’t....
Large-scale cluster management at Google with Borg
Monitoring
Monitoring in a nutshell

- An incredibly simple, complex problem
- “Collecting, processing, aggregating and displaying real-time quantitative data about a system, such as query counts and types, error counts and types, processing times and server lifetimes.”
- Analyze long term trends
- Build dashboards
- Alerting & Anomaly detection
- 4 Golden Signals: Latency, Traffic, Errors & Saturation
Monitoring Implementation & Example

```java
handleRequest(Request r) {
    REQUEST_COUNT++;
    // ...
    if (Error)
        ERROR_COUNT++;
}
```

Another system scrapes the value of the globals for each task
Computing Metrics

For a task:

\[ \text{QPS} = \Delta \text{REQUEST\_COUNT} \]

\[ \text{EPS} = \Delta \text{ERROR\_COUNT} / \Delta \text{REQUEST\_COUNT} \]

For a Cluster:

\[ \text{QPS} = \Delta \text{AGGREGATE\_REQUEST\_COUNT} \]

\[ \text{ERROR\_RATE} = \]
\[ \Delta \text{AGGREGATE\_ERROR\_COUNT} / \Delta \text{AGGREGATE\_REQUEST\_COUNT} \]
What about latency?

Use buckets!

LATENCY_COUNT[0..99]++;
LATENCY_COUNT[100..199]++;
LATENCY_COUNT[200..299]++;
LATENCY_COUNT[300..399]++;
...
LATENCY_COUNT[10000..∞]++;
What about QPS per API method?

Two options. One variable per method, or one variable with a method dimension.

```c
REQUEST_COUNT[ 'WRITE' ]++;  
REQUEST_COUNT[ 'READ' ]++;  
REQUEST_COUNT[ 'DELETE' ]++;  
```
Now we have metrics. What do we do with them?
Alerting

if( ERROR_RATE > .05 : 10m )
    WakeUpPaul();

if( QPS < 10 : 5m )
    WakeUpPaul();

What if the alerting fails?

What should your thresholds be?
More Definitions!

- SLO = Service Level Objective
- SLA = Service Level Agreement
- SLI = Service Level Indicator
- “nines” = .999...
<table>
<thead>
<tr>
<th>Nines</th>
<th>Downtime per Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Nine = .9</td>
<td>1 Nine =</td>
</tr>
<tr>
<td>2 Nines = .99</td>
<td>2 Nines =</td>
</tr>
<tr>
<td>3 Nines = .999</td>
<td>3 Nines =</td>
</tr>
<tr>
<td>4 Nines = .9999</td>
<td>4 Nines =</td>
</tr>
<tr>
<td>5 Nines = .99999</td>
<td>5 Nines =</td>
</tr>
</tbody>
</table>
## SLO with Nines

<table>
<thead>
<tr>
<th>Nines</th>
<th>Downtime per Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Nine = .9</td>
<td>1 Nine = 36.53 days</td>
</tr>
<tr>
<td>2 Nines = .99</td>
<td>2 Nines =</td>
</tr>
<tr>
<td>3 Nines = .999</td>
<td>3 Nines =</td>
</tr>
<tr>
<td>4 Nines = .9999</td>
<td>4 Nines =</td>
</tr>
<tr>
<td>5 Nines = .99999</td>
<td>5 Nines =</td>
</tr>
</tbody>
</table>
# SLO with Nines

<table>
<thead>
<tr>
<th>Nines</th>
<th>Downtime per Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Nine = .9</td>
<td>1 Nine = 36.53 days</td>
</tr>
<tr>
<td>2 Nines = .99</td>
<td>2 Nines = 3.65 days</td>
</tr>
<tr>
<td>3 Nines = .999</td>
<td>3 Nines =</td>
</tr>
<tr>
<td>4 Nines = .9999</td>
<td>4 Nines =</td>
</tr>
<tr>
<td>5 Nines = .99999</td>
<td>5 Nines =</td>
</tr>
<tr>
<td>Nines</td>
<td>Downtime per Year</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1 Nine</td>
<td>1 Nine = 36.53 days</td>
</tr>
<tr>
<td>2 Nines</td>
<td>2 Nines = 3.65 days</td>
</tr>
<tr>
<td>3 Nines</td>
<td>3 Nines = 8.77 hours</td>
</tr>
<tr>
<td>4 Nines</td>
<td>4 Nines =</td>
</tr>
<tr>
<td>5 Nines</td>
<td>5 Nines =</td>
</tr>
</tbody>
</table>
### SLO with Nines

<table>
<thead>
<tr>
<th>Nines</th>
<th>Downtime per Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Nine = .9</td>
<td>1 Nine = 36.53 days</td>
</tr>
<tr>
<td>2 Nines = .99</td>
<td>2 Nines = 3.65 days</td>
</tr>
<tr>
<td>3 Nines = .999</td>
<td>3 Nines = 8.77 hours</td>
</tr>
<tr>
<td>4 Nines = .9999</td>
<td>4 Nines = 52.60 minutes</td>
</tr>
<tr>
<td>5 Nines = .99999</td>
<td>5 Nines =</td>
</tr>
<tr>
<td>Nines</td>
<td>Downtime per Year</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1 Nine = .9</td>
<td>1 Nine = 36.53 days</td>
</tr>
<tr>
<td>2 Nines = .99</td>
<td>2 Nines = 3.65 days</td>
</tr>
<tr>
<td>3 Nines = .999</td>
<td>3 Nines = 8.77 hours</td>
</tr>
<tr>
<td>4 Nines = .9999</td>
<td>4 Nines = 52.60 minutes</td>
</tr>
<tr>
<td>5 Nines = .99999</td>
<td>5 Nines = 5.26 minutes</td>
</tr>
</tbody>
</table>
Keeping Systems Running

“spes non consilium est”
On-call Rotations

- Systems run 24 x 7 and need support 24 x 7
- Can be mandatory or voluntary
- Can be paid or unpaid
- Have different SLOs for response
Site Reliability Engineers

- **Engineers** who build systems that support running, scaling, releasing, monitoring and configuring systems.
- Sysadmin as a software problem
- People manually doing things -> People using tools to do things -> Machines using tools to do things -> Problem never happens
- Reduce Toil
- Minimize Risk
- Standardize production
Things will still break. The unexpected will still happen.

- Config pushes
- Fires
- Hunters, wildlife, etc
- Code changes
- Hardware failure
- Solar rays
- Disaster Recovery Testing
Q) What is the Most Important Lesson we have learned for Building & Maintaining the Building Blocks of the Web
A) Psychological Safety

Psychological safety is a shared belief that the team is safe for interpersonal risk taking. It can be defined as "being able to show and employ one's self without fear of negative consequences of self-image, status or career" - Wikipedia
Blameless Post-mortems

- What went wrong?
- What was the impact?
- What protections could there of been?
- What enabled a human to believe that was the correct action to take?
- What could of prevented that?