Runtime Analysis and Testing in the Cloud

Dr. Wolfgang Grieskamp
Staff Software Engineer, Google USA

CREST Workshop, May 20th, 2012

About me

- < 2000: Researcher and Lecturer at Technical University of Berlin
- 2000-2006: Senior Researcher, Microsoft Research
- 2007-2011: Principal Architect, Microsoft Windows Interoperability Team, Server and Cloud division
- Since 4/2011: Staff Engineer, Google+ platform and tools, Google

- **DISCLAIMER**: *This talk does not necessarily represent Google's opinion or direction.*

About this talk

Will talk about:

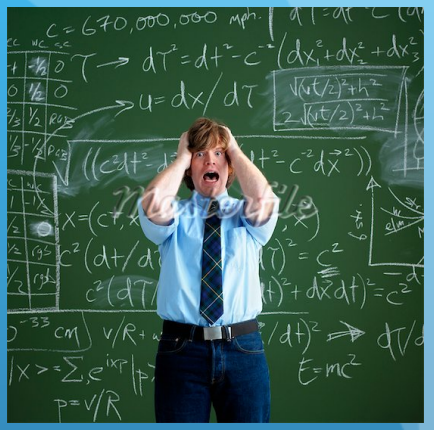
- How Google monitors and tests Cloud software
- Quick pitch how Google uses the Cloud itself for development

Will assume:

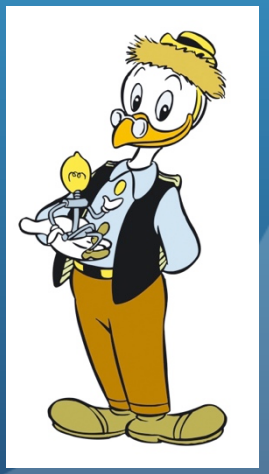
- You know something about software engineering and about Cloud computing

My Viewpoint

- As a researcher who tries to identify open problems (and none-problems!)



- As an engineer who tries to understand and improve the process.



What is Cloud Computing?

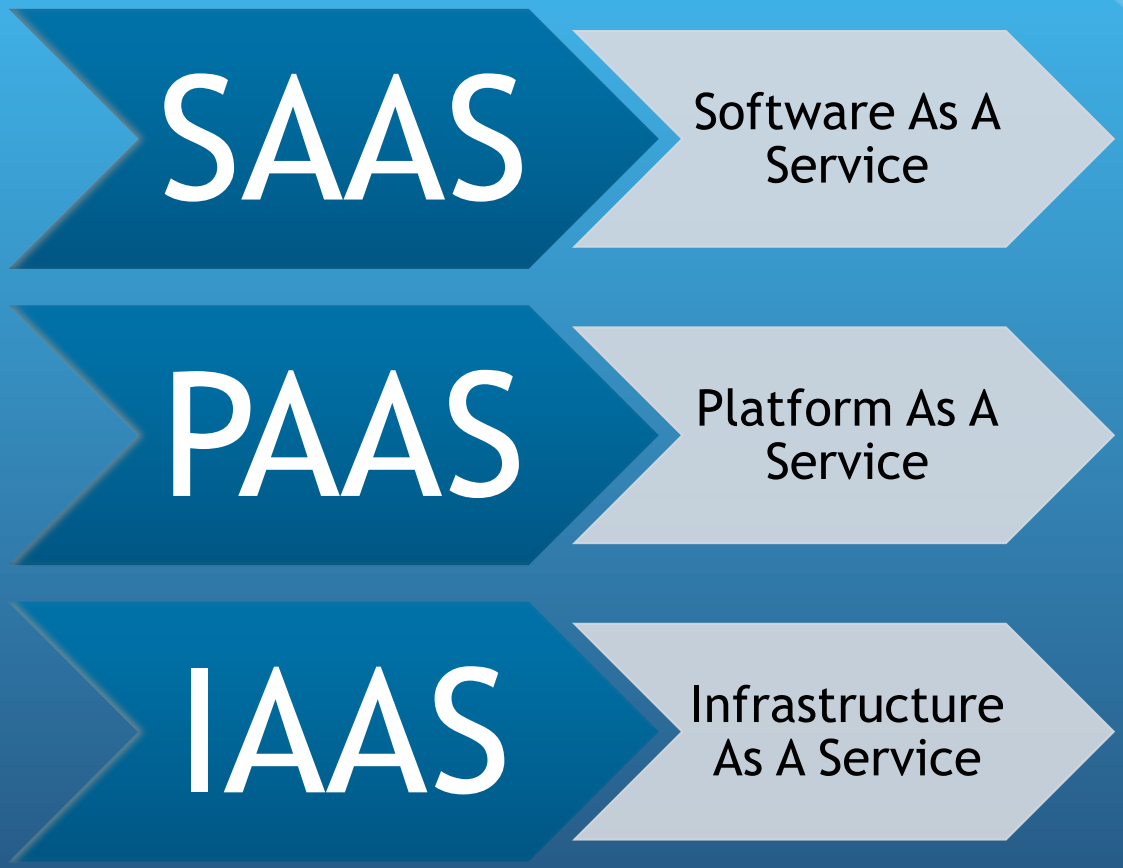
From Wikipedia, the free encyclopedia

Cloud computing is the **delivery of computing as a service** rather than a product,

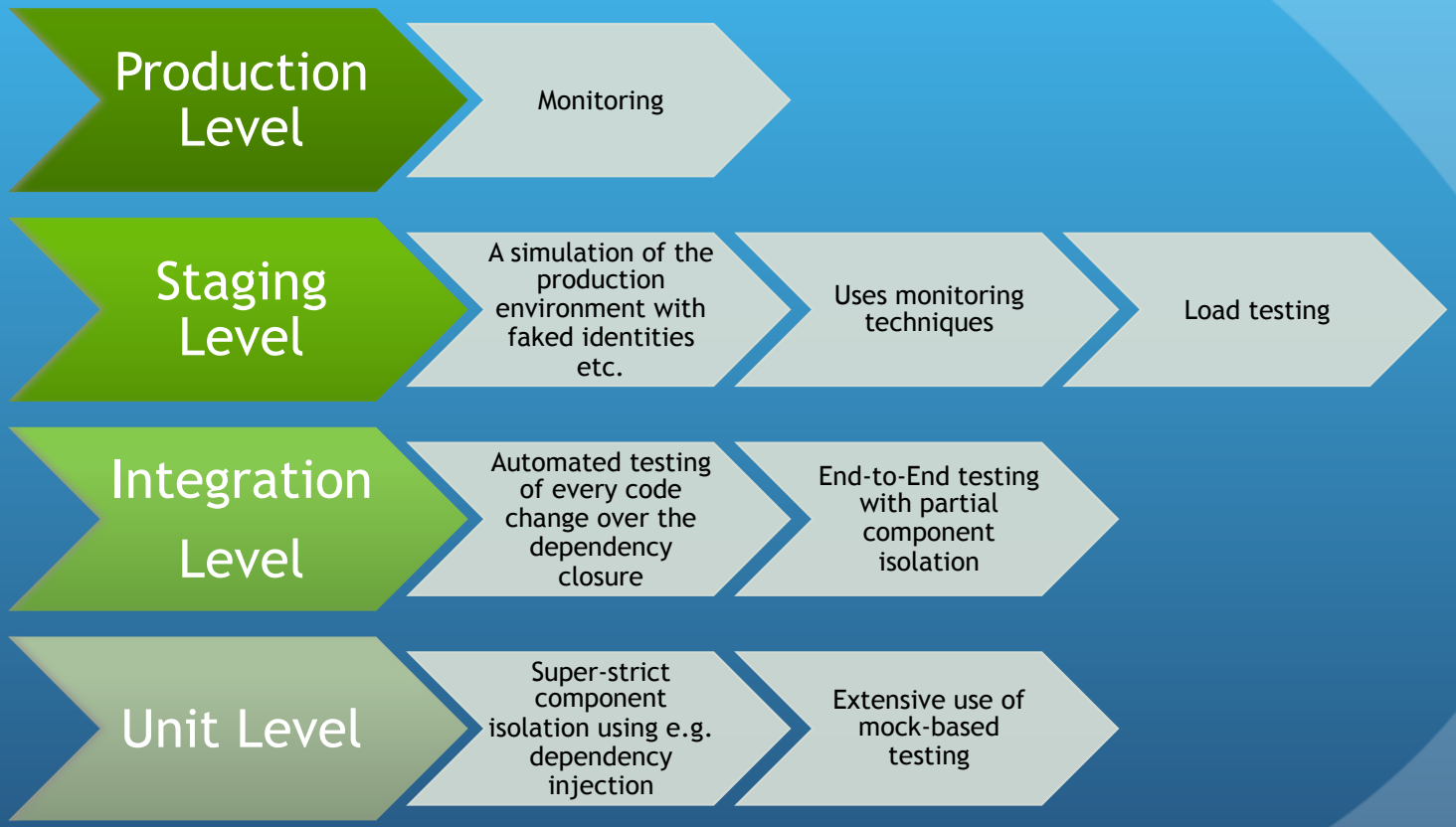
whereby **shared resources, software and information are provided** to computers and other devices

as a **utility** (like the electricity grid) over a network (typically the Internet).

Cloud Stack


SAAS
Software As A Service
PAAS
Platform As A Service
IAAS
Infrastructure As A Service

Runtime Analysis and Testing @ Google

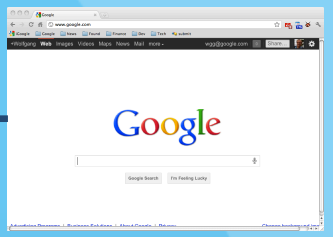


Monitoring and Testing

What the heck is the difference?

- In testing…
  - we *simulate (mock)* the environment (aka user)
  - we don't care as much about performance overhead
- In monitoring…
  - we are interested mostly in general health not detailed functionality (assumed its already tested)
  - we use stochastic methods more frequently
- Otherwise many things similar.

Anatomy of a Data Center



Data Center A

Controller

Controller

Server

Server

Server

Server

Server

...

Storage

Storage

Storage

Storage

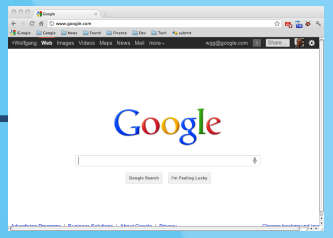
Storage

Data Center B

.....

Note: abstracted and simplified

Anatomy of a Server



Data Center A

Server (VM)

Controller

Controller

Controller

Server

Server

Job

Job

Job

...

Monitor

Monitor

Monitor

Storage

Storage

Storage

Storage

Alert

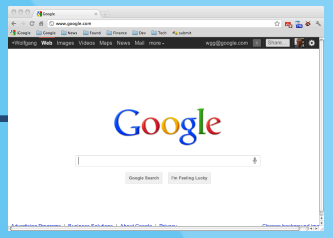
Logs

Data Center B

.....

Note: abstracted and simplified

Anatomy of a Service



Data Center A

Service (across Servers)

Controller

Job

Job

Server

Job

Job

Job

Server

...

Job

Storage

Storage

Storage

Data Center B

.....

Note: abstracted and simplified

Monitoring Types @ Google

- Black Box Monitoring
- White Box Monitoring
- [Log Analysis]

Black Box Monitoring
How its done @Google

- Frequently send requests and analyze the response
  - Possible because server jobs are ‘stateless’ and always input enabled
- If failure rate over a certain time interval exceeds a given ratio, raise an alert and page an engineer
  - Engineers aim for minimizing paging and avoiding false positives

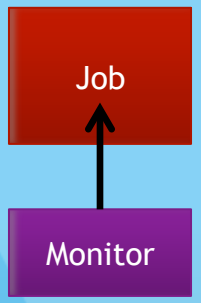
Black Box Monitoring: How its done @ Google (cont.)

- There are rule based languages for defining request/ responses. Each rule:
  - Synthesizes an HTTP request
  - Analyzes the response using a regular expression
  - Specifies frequency and allowed failure ratio
- Rules are like tests: a simple trigger and a simple response analysis
- Monitors can be also custom code

Black Box Monitoring: How is it doing?

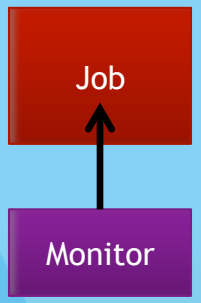
- Is the ‘stateless’ hypothesis feasible?
  - Yes, as these are health tests, state can be ignored
- What is the relation to testing?
  - In theory very similar, only that the environment is not mocked.
  - In practice uses quite different frameworks/languages
- What about service/system level monitoring?
  - Its only about *one* job.
  - Doesn’t give failure root cause (it only measures a *symptom*)

White-Box Monitoring How its done @Google

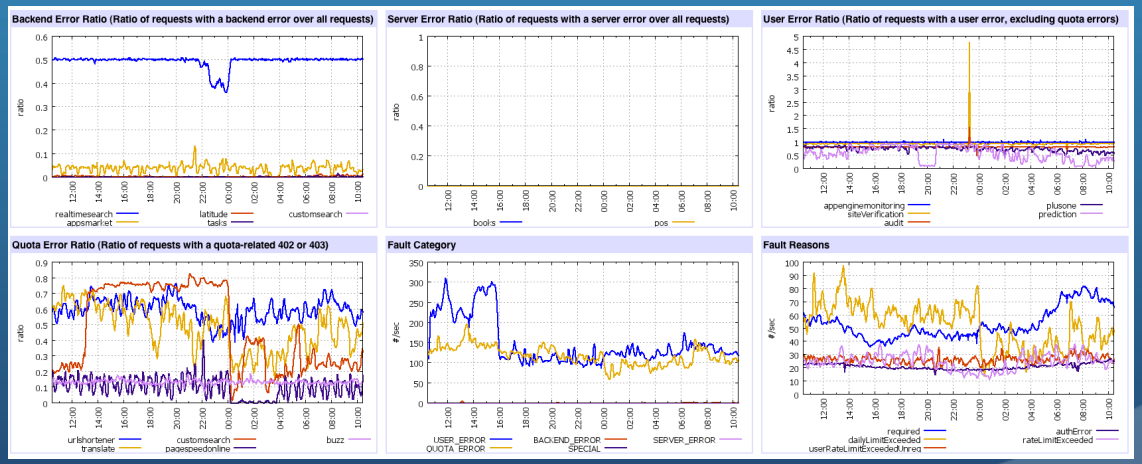


- Server exports collection of probe points (variables)
  - Memory, # RPCs, # Failures, etc.
- Monitor collects *time series* of those values and computes functions over them
- Dashboards prepare information graphically
- Mostly used for diagnosis by humans

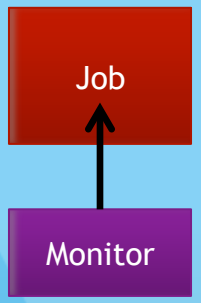
White-Box Monitoring: How its done @ Google (cont.)



- Declarative language for time series computations
- Collects samples from the server by memory scraping
- Merging of similar data from multiple servers running the same job
- Rich support for diagram rendering in the browser

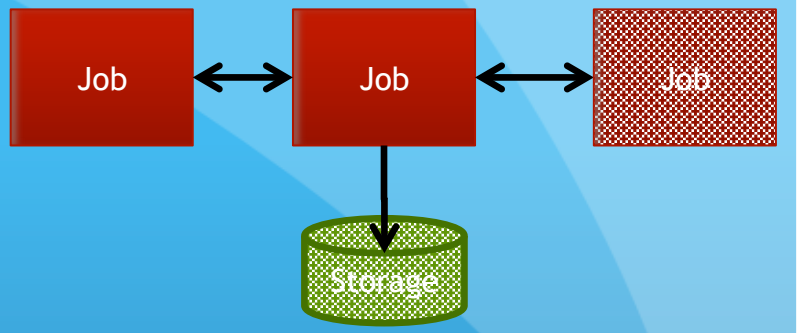


White-Box Monitoring: How is it doing?



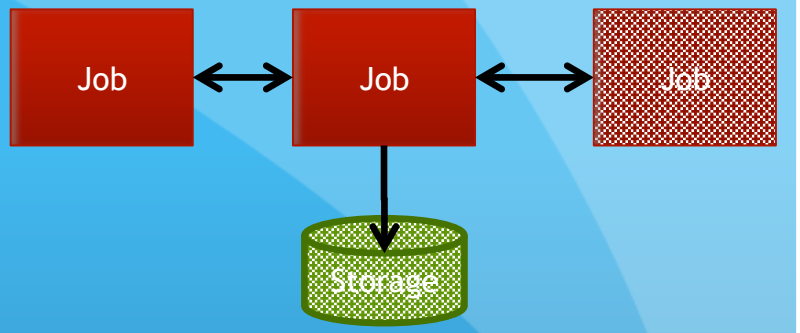
- Design for monitorability/testability?
  - Its already ubiquitous throughout, since software engineers are themselves on-call…
- Distributed collection/network load?
  - Not really an issue because it's sample based
- Relation to testing?
  - Same as with black-box – should be a common framework.
- Automatic root cause analysis and self-repair?
  - Current systems mostly build for human analysis and repair.
  - Self-repair would be a big thing.

Integration Testing: How its done @Google



- Two or more components are plugged together with a partially mocked environment
- The environment provides stimuli and checks expectations
- Usually runs on a single machine
- Can be deployed to the cloud for large scale testing

Integration Testing
How is it doing?



- Integration test are often ‘flaky’ (unreliable)
- Difficulty to construct mocked component’s precise behavior (its more than a simple mock in a unit test)
- Difficulty to synthesize mocked component’s initial state (it may have a complex state)
- Potential solution: model-based testing and simulation

Exploiting the Cloud for Development

Idle Resources

Peak demand problem: as with other utilities, the cloud must have capacity to deal with peak times: 7am, 7pm, etc.

- Huge amounts of idle computing resources available in the DCs outside of those peak times
- Literally hundreds of VMs may be available for a single engineer on a low-priority job base

➔Game changer for software development tools

Using the Cloud for Dev @ Google

- Distributed/parallel build
  - Every engineer can build all of Google's code + third party open source code in a matter of minutes (sequential build would take days)
  - Works by constructing the dependency graph than using map/reduce technology
- Distributed/parallel test
  - Changes on the code base are continuously tested against all dependent targets once submitted
  - Failures can be tracked down very precisely to the given change which have introduced them
- Check out http://google-engtools.blogspot.com/ for details

Conclusions

- The Cloud brings new challenges for runtime analysis and testing.
  - Many of them are adequately solved – others wait for improvements.
- The Cloud brings new opportunities for software development tools.