• Software Plumber, mix of Enterprise and Consumer
  • 18 years writing software, backend guy with a taste for javascript
  • 2 y Accenture (Notes guru), 3 y Netscape/AOL (Servers, Portals), 5 y Sun (ecommerce, blogs, Portals, feeds, open source)
  • 6 years at Google, API guy (first hired, helped start the team)
  • Adwords, Checkout, Social, HTML5, Cloud
P@ & Military Software

1993 - Software metrics Dod-Std-2167A

1994 - C3I

MINISTÈRE DE LA DÉFENSE

DGA

Fig. 1 semantics information complexity
Architecture Changes: 80’s Client-Server
Architecture Changes: 90’s Web
Architecture Changes: 2010’s Cloud, HTML5, Mobile
Hype warning: Cloudy, with a chance of real innovation

Source: Gartner (August 2009)
Cloud started at Consumer websites solving their needs

- Google, Amazon, Yahoo, Facebook, Twitter
- Large Data Sets
- Storage Capacity growing faster than Moore’s Law
- Fast Networks
- Horizontal -> Vertical scalability
- Open Source Software
- Virtualization
- Cloud is a productization of these infrastructures
  - Public Clouds Services: Google, Amazon
  - Open Source Software: Hadoop, Eucalyptus, Cloud Foundry
Cloud, according to my daughter
Cloud Computing Categories

- IaaS
- PaaS
- SaaS

Source: Gartner AADI Summit Dec 2009
Cloud Computing Categories

IaaS

PaaS

SaaS

Source: Gartner AADI Summit Dec 2009
Cloud Computing Categories

SaaS

PaaS

IaaS

Source: Gartner AADI Summit Dec 2009
Infrastructure culture

- Larry and Serguey’s 1998 paper "The Anatomy of a Large-Scale Hypertextual Web Search Engine"
  - http://infolab.stanford.edu/~backrub/google.html

- Other Google Research papers since then
  - http://research.google.com/pubs/papers.html

- Build on the shoulders of giants

- Custom stack made of standards parts: machines, linux, servers

- Standard infrastructure: sharding, GFS, MapReduce, BigTable

- Google App Engine: easy cloud, for Googlers and others developers

- Standard languages: c/c++, java, python

- Horizontal scalability: parallel and asynchronous whenever possible
Programming the Cloud – The Google Way

• Fault tolerant distributed storage: Google File System
• Distributed shared memory: Bigtable
• New programming abstractions: MapReduce
• Domain Specific Languages: Sawzall
Fault Tolerant Distributed Disk Storage: GFS

- Data replicated 3 times. Upon failure, software re-replicates.
- Master: Manages file metadata. Chunk size 64 MB.
- Optimized for high-bandwidth sequential read / writes
- Clusters > 5 PB of disk

Distributed Shared Memory: Bigtable

- Sparse, distributed, persistent, multidimensional, sorted
- Not a relational database (RDBMS): no schema, no joins, no foreign key constraints, no multi-row transactions
- Each row can have any number of columns, similar to a dictionary data structure for each row.
- Basic data types: string, counter, byte array
- Accessed by row key, column name, timestamp
- Data split into tablets for replication
- Largest cells are > 700TB

http://research.google.com/archive/bigtable-osdi06.pdf
# Datastore layers

<table>
<thead>
<tr>
<th></th>
<th>Complex queries</th>
<th>Entity Group Transactions</th>
<th>Queries on properties</th>
<th>Key range scan</th>
<th>Get and set by key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Megastore</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Bigtable</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Megastore API

• “Give me all rows where the column ‘name’ equals ‘ikai’”
• “Transactionally write an update to this group of entities”
• “Do a cross datacenter write of this data such that reads will be strongly consistent” (High Replication Datastore)
Programming Abstraction: MapReduce

- Represent problems as Map and Reduce step (inspired by functional programming)
- Distribute data among many machines, execute same computation at each machine on its dataset
- Infrastructure manages parallel execution
- Open source implementation: Hadoop

```
map(in_key, data) → list(key, value)
reduce(key, list(values)) → list(out_data)
```

http://research.google.com/archive/mapreduce.html
Language for Parallel Log Processing: Sawzall

- Commutative and associative operations allow parallel execution and aggregation
- Language avoids specifying order by replacing loops with quantifiers (constraints)

```plaintext
count: table sum of int;
total: table sum of float;
x: float = input;
emit count <- 1;
emit total <- x;
```

http://labs.google.com/papers/sawzall.html
Internet as a Platform: The Challenges

Architect’s Dream

- Loosely coupled
- Extensible
- Standards-based
- Fault tolerant
- Unlimited computing power
- Ubiquitous
<table>
<thead>
<tr>
<th>Architect’s Dream</th>
<th>Developer’s Nightmare</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Loosely coupled</td>
<td>• NO Call Stack</td>
</tr>
<tr>
<td>• Extensible</td>
<td>• NO Transactions</td>
</tr>
<tr>
<td>• Standards-based</td>
<td>• NO Promises</td>
</tr>
<tr>
<td>• Fault tolerant</td>
<td>• NO Certainty</td>
</tr>
<tr>
<td>• Unlimited computing power</td>
<td>• NO Ordering Constraints</td>
</tr>
<tr>
<td>• Ubiquitous</td>
<td></td>
</tr>
</tbody>
</table>
New Game Rules

ACID (before)

ACID (today)
New Game Rules

ACID (before)

• Atomic

ACID (today)
New Game Rules

ACID (before)

• Atomic
• Consistent

ACID (today)
New Game Rules

ACID (before)

- Atomic
- Consistent
- Isolated

ACID (today)
New Game Rules

ACID (before)

- Atomic
- Consistent
- Isolated
- Durable

ACID (today)
New Game Rules

ACID (before)
- Atomic
- Consistent
- Isolated
- Durable

ACID (today)
- Associative
New Game Rules

ACID (before)
- Atomic
- Consistent
- Isolated
- Durable

ACID (today)
- Associative
- Commutative

© 2008 Google, Inc. All rights reserved,
New Game Rules

ACID (before)

- Atomic
- Consistent
- Isolated
- Durable

ACID (today)

- Associative
- Commutative
- Idempotent

© 2008 Google, Inc. All rights reserved,
<table>
<thead>
<tr>
<th>ACID (before)</th>
<th>ACID (today)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>Associative</td>
</tr>
<tr>
<td>Consistent</td>
<td>Commutative</td>
</tr>
<tr>
<td>Isolated</td>
<td>Idempotent</td>
</tr>
<tr>
<td>Durable</td>
<td>Distributed</td>
</tr>
</tbody>
</table>
New Game Rules

<table>
<thead>
<tr>
<th>ACID (before)</th>
<th>ACID (today)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>Associative</td>
</tr>
<tr>
<td>Consistent</td>
<td>Commutative</td>
</tr>
<tr>
<td>Isolated</td>
<td>Idempotent</td>
</tr>
<tr>
<td>Durable</td>
<td>Distributed</td>
</tr>
</tbody>
</table>

Predictive
Accurate

Flexible
Redundant
Starbucks Does not Use 2-Phase Commit Either

• Start making coffee before customer pays
• Reduces latency
• What happens if…
Starbucks Does not Use 2-Phase Commit Either

- Start making coffee before customer pays
- Reduces latency
- What happens if…

Customer rejects drink

Coffee maker breaks

Customer cannot pay
Starbucks Does not Use 2-Phase Commit Either

• Start making coffee before customer pays
• Reduces latency
• What happens if…

Customer rejects drink ———> Remake drink

Coffee maker breaks

Customer cannot pay
Starbucks Does not Use 2-Phase Commit Either

• Start making coffee before customer pays
• Reduces latency
• What happens if…

Customer rejects drink → Remake drink
Coffee maker breaks → Refund money
Customer cannot pay
Starbucks Does not Use 2-Phase Commit Either

• Start making coffee before customer pays
• Reduces latency
• What happens if…

Customer rejects drink ➔ Remake drink

Coffee maker breaks ➔ Refund money

Customer cannot pay ➔ Discard beverage
Starbucks Does not Use 2-Phase Commit Either

• Start making coffee before customer pays
• Reduces latency
• What happens if…

Customer rejects drink   Remake drink
                    Retry
Coffee maker breaks  Refund money
Customer cannot pay  Discard beverage
Starbucks Does not Use 2-Phase Commit Either

- Start making coffee before customer pays
- Reduces latency
- What happens if…

Customer rejects drink → Remake drink
  - Retry
Coffee maker breaks → Refund money
  - Compensation
Customer cannot pay → Discard beverage
Starbucks Does not Use 2-Phase Commit Either

- Start making coffee before customer pays
- Reduces latency
- What happens if...

Customer rejects drink ➔ Remake drink
  - Retry

Coffee maker breaks ➔ Refund money
  - Compensation

Customer cannot pay ➔ Discard beverage
  - Write-off
Commoditization of distributed computing concepts & tools

- Languages: Erlang concepts -> Go, Scala
- NoSQL Zoo: BigTable, HBase, MongoDB, Redis, Cassandra
- Map/Reduce: Apache Hadoop
- Paxos, Eventual Consistency, CAP Theorem
- REST, statelessness, idempotency
Economic Drivers

• Proportion of electricity in cost of computing

• Product -> Service

• Economies of Scale

• Moore’s Law

• Pay as you go utility model
Cultural Drivers

• Expectations of corporate IT customers have changed

• Consumerization of IT

• Consumer apps more and more like fashion

• Technology achieves ubiquity by disappearing
Access from Anywhere
Scales Up, Scales Down, with Demand
Innovation Not Administration
Cultural Drivers: Agility

- Waterfall -> Agile methodologies
- Cloud enables an Agile culture, driver for innovation
Fail often, fail quickly, and learn
Fail often, fail quickly, and learn

- Risk taking/Experimentation is encouraged
  - http://blog.red-bean.com/sussman/?p=96

- “Do not be afraid of day-to-day failures — learn from them. (As they say at Google, “don’t run from failure — fail often, fail quickly, and learn.”) Cherish your history, both the successes and mistakes. All of these behaviors are the way to get better at programming. If you don’t follow them, you’re cheating your own personal development.”

- Ben Collins-Sussman (Subversion, code.google.com)
Agile Development Processes
Agile Development Processes

• Influences from XP, Agile, Scrum
• Code reviews
• Test Driven Development: Testing on the Toilets program and blog
• Many internal development tools: Mondrian recently open sourced
• Changed the meaning of beta
• Teams co-located: 3-15 people, 4/cubicle, all close to each other
• International offices: manage whole projects, avoid coordination costs
Open Source Culture
Open Source Culture

• Open Source Program Office

• Summer of Code

• Open sourcing parts of Google code
  • http://code.google.com/

• Making the web better: GWT, Gears, OpenSocial, Android
API Culture
API Culture

• Bill Joy: "Innovation happens elsewhere"

• From 3 to 62 APIs in 3 years

• Maps on websites

• Friend Connect: all sites can become social
  • http://code.google.com/ for the list

• Build an ecosystem around the APIs (my job)

• User's choice: get their data out
Users should be able to control the data they store in any of Google's products. Our team's goal is to make it easier to move data in and out.
Google Cloud Products
Google's Cloud Offerings

1. Google Apps
2. Third party Apps: Google Apps Marketplace
3. ________

- IaaS
- PaaS
- SaaS

- Google App Engine
- Google Storage Prediction API
- BigQuery

Thursday, May 26, 2011
Google's Cloud Offerings

Your Apps

1. Google Apps
2. Third party Apps: Google Apps Marketplace
3. ________

IaaS

PaaS

SaaS

Google App Engine

Google Storage Prediction API
BigQuery
How Google Apps Adds Value

Productivity and Innovation
Realtime collaboration, constant updates, new features

Platform Independence
Work anywhere from any computer or mobile device

Reduced IT Complexity
Least complex, least expensive to license and manage
How Google Apps Adds Value

Security and Availability
Same uptime and infrastructure used for Google products

Built-in Enterprise Security Features
2-Factor Authentication, Single Sign On, Reporting Tools
How Google Apps Adds Value

Security and Availability
Same uptime and infrastructure used for Google products

Built-in Enterprise Security Features
2-Factor Authentication, Single Sign On, Reporting Tools

A Toolbox of Administrative APIs
Reporting, Compliance, Identity Management and more...
Why Google App Engine?

- Easy to build
- Easy to maintain
- Easy to scale
Cloud Development in a Box

• Downloadable SDK
• Application runtimes
  ○ Java, Python
• Local development tools
  ○ Eclipse plugin, AppEngine Launcher
• Specialized application services
• Cloud based dashboard
• Ready to scale
• Built in fault tolerance, load balancing
Specialized Services

- Memcache
- Datastore
- URL Fetch
- Mail
- XMPP
- Task Queue
- Images
- Blobstore
- User Service
Language Runtimes
App Engine Growth

- App Engine Launch
- Python
- Datastore
- Memcache
- logs export
- Batch write/read
- Https
- Status-Dashboard
- Java
- DB Import
- cron
- Task Queues
- XMPP
- incoming email
- Blobstore
- Appstats
- Mapper
- Multitenancy
- Instance Console
- Always On
- hi-perf imag
- 10 min tasks
- Hi-Replication
- Channel API
- Files API
- Remote API
- Prosp Search

Thursday, May 26, 2011
By the Numbers

200,000 Active apps per week
By the Numbers

1.5B Pageviews per day
Some App Engine Partners

Google Cloud In the Government sector
Google is Committed to Meeting the Add'l Challenges of Governments

Google Apps for Government
• Government only cloud in dedicated CONUS facilities
• Government pricing (GSA Schedule 70)

ATO Issued by GSA
• Certification & Accreditation based on Federal Information Systems Management Act (FISMA)
• FIPS 199 Moderate baseline

Leader in Public Sector Cloud Computing
• General Services Administration, USAID, Dept of Energy, Colorado, New Mexico, Orlando, Los Angeles, Treasury, Multnomah County, Smithsonian, Tennessee Valley Authority, ACUS, MACPAC, Wyoming
Select DoD Customers

Requirements
- Provide common platform for non-classified collaboration

The Solution
- Implement Google Apps (eushare.org) to support unclassified communication and coordination across 51 countries throughout the Area of Responsibility
- Used during planning, execution, and after-action reporting of X24Europe, a recent global exercise

Requirements
- Provide common platform to support HA/DR mission area

The Solution
- Implement Google Apps (InRelief.org) to support cross-country, cross-language collaboration; Includes consolidation of social networking signals

Requirements
- Provide messaging platform for families to connect with deployed service members

The Solution
- Garrison Commander's toolkit for communicating to troops and their families.
Factors to consider for picking a Cloud

- Price
- Type: IaaS, PaaS, SaaS
- Type of task: Apps, Big Data
- Public/Private/Hybrid
- Lock-In: Standards, Open Source
Issues to solve

• Cloud Interop: lack of standard

• Replication of Data across multiple Clouds

• Data privacy/integrity
  • encryption at rest
  • data auditing

• Trust, Culture of agility
Google Cloud Clients

- Chrome, HTML5
- ChromeBook, Device as a service $28/user/month
- Android: phone and tablets