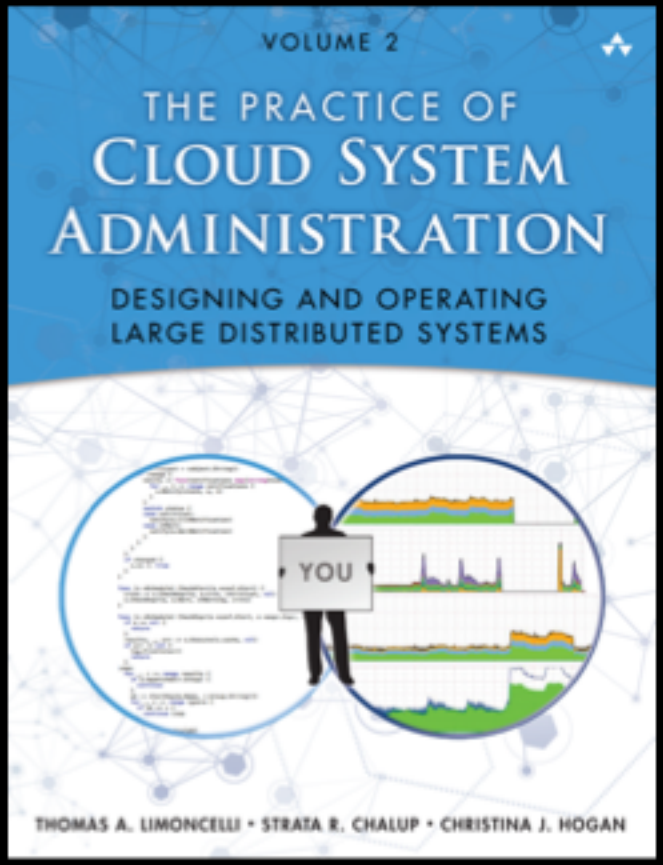
Presented at
Back Bay LISA

January 14, 2015

Cambridge, MA

http://bblisa.org

Radical Ideas from The Practice of Cloud System Administration



Tom Limoncelli, SRE

StackExchange.com

the-cloud-book.com

@YesThatTom

Who is Tom Limoncelli?

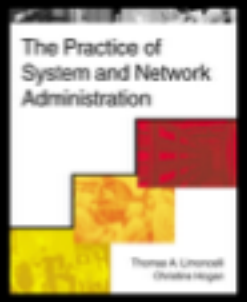
Sysadmin since 1988

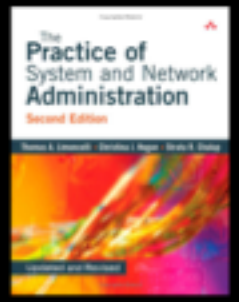
Worked at Google, Bell Labs plus many smaller companies.

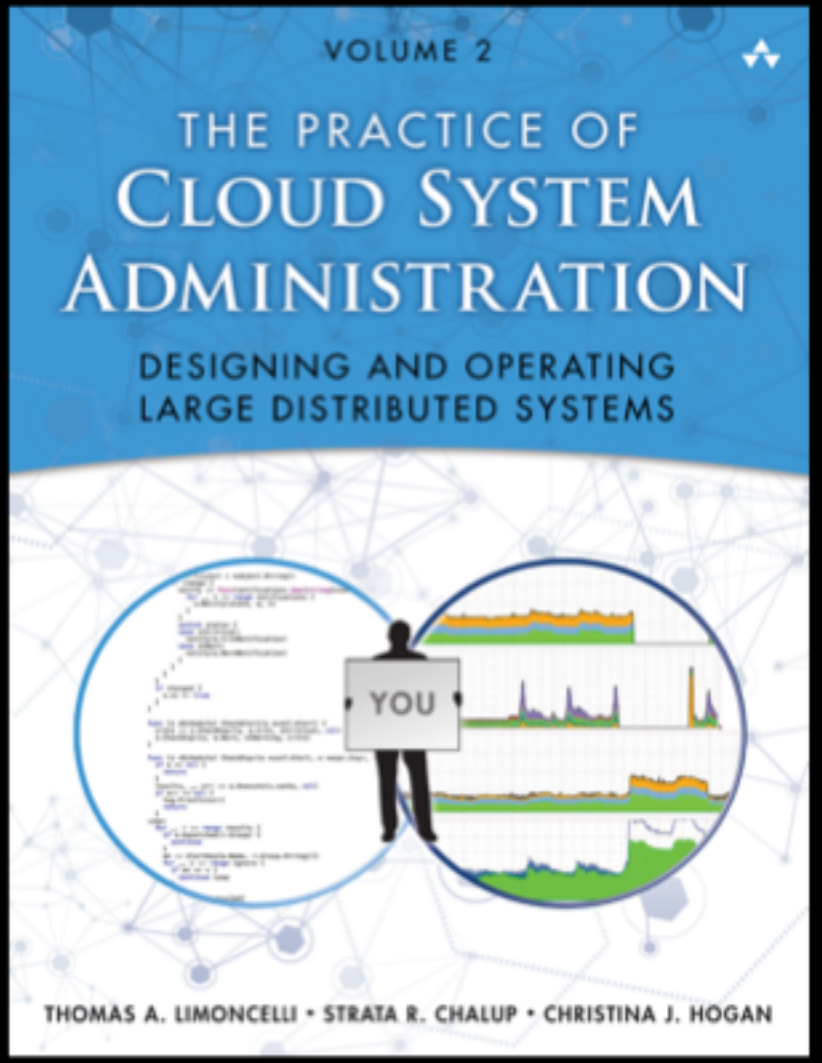
SRE at Stack Exchange, Inc
serverfault.com / stackoverflow.com

Blog: **EverythingSysadmin.com**

Twitter: **@YesThatTom**



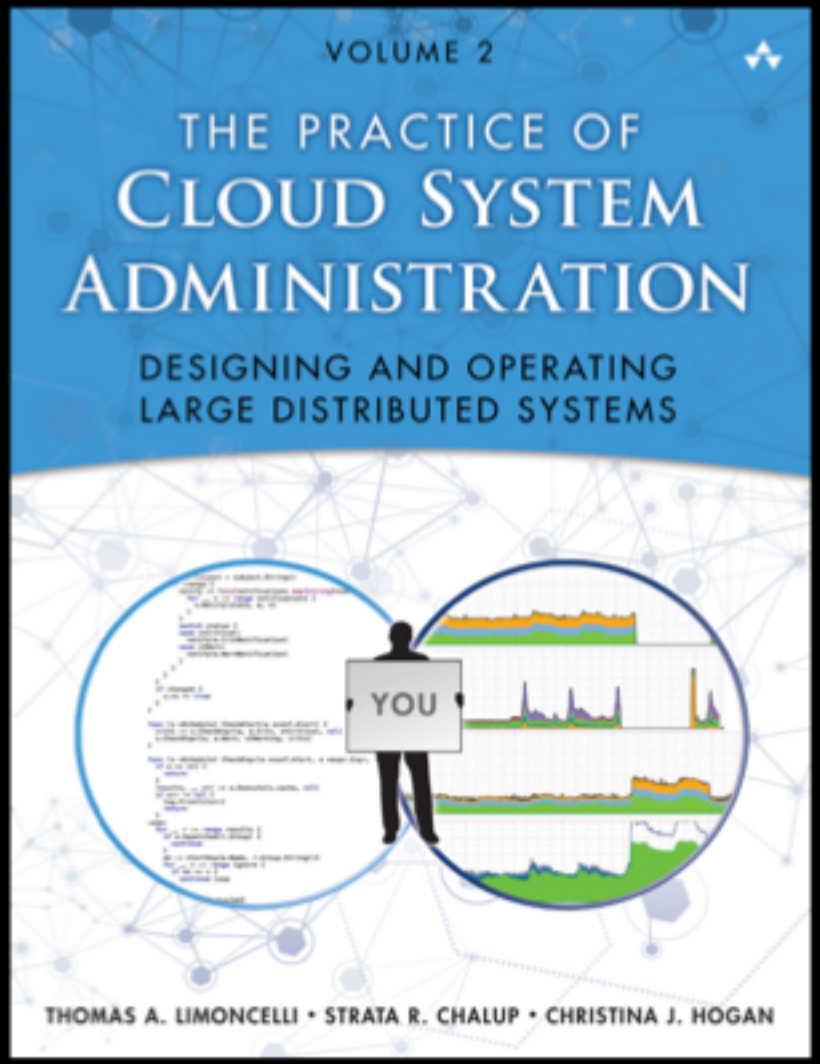




Thomas A. Limoncelli is internationally recognized author, speaker, and system administrator with 20+ years of experience at companies like Google, Bell Labs and StackExchange.com

Strata R. Chalup has 25+ years experience in Silicon Valley focusing on IT strategy, best-practices, and scalable infrastructures at firms including Apple, Sun, Cisco, McAfee, and Palm.

Christina J. Hogan has 20+ years experience in system administration and network engineering, from Silicon Valley, to Italy, and Switzerland. She has a Masters in CS, a PhD in Aeronautical Engineering and has been part of a Formula 1 racing team.



- 2012 April: Started writing
- 2014 April: Chapters Done
- 2014 May/June: Copyediting
- 2014 July/Aug: Layout
- 2014-Sept-14: SHIPPED!

April
2012

Sept
2014

The Cloud

The Cloud

The
Cloooooouud

The Cloud!!!!!!

The Cloud!!1!

We
<heart>
The Cloud

The cloud solves all problems.

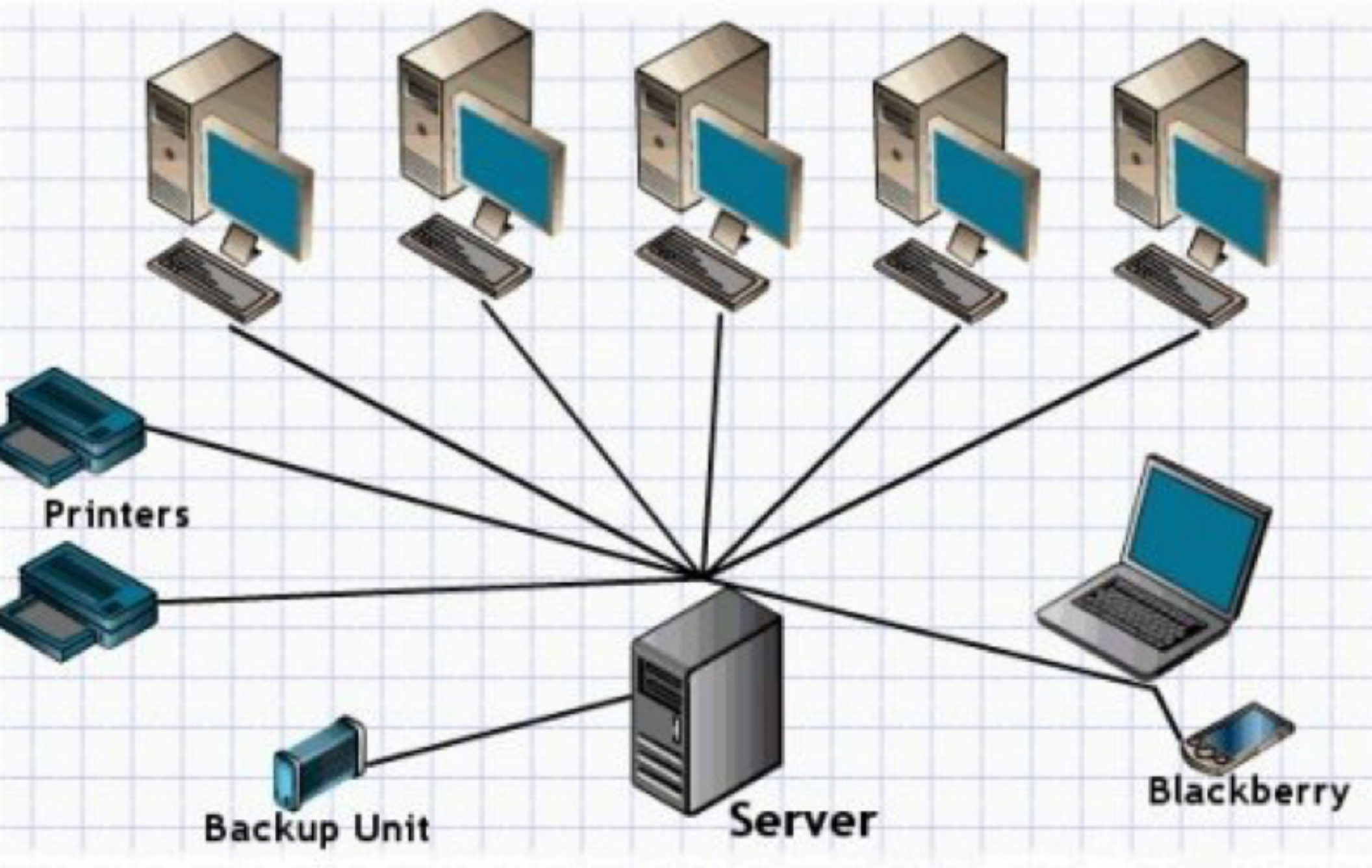
Cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud.

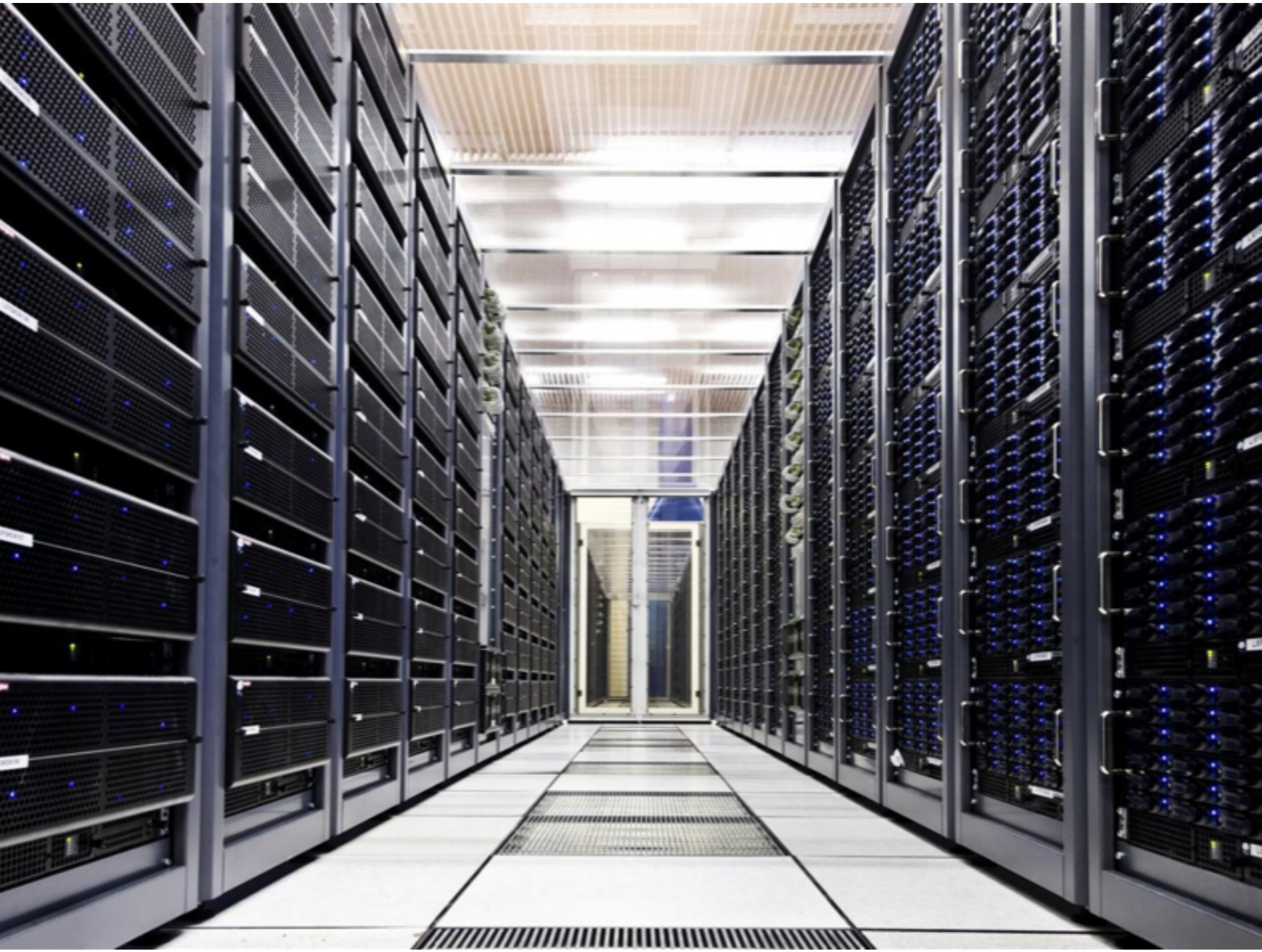
Distributed Computing

COURTESY NATIONAL MUSEUM OF COMPUTING/EXPRESS&STAR

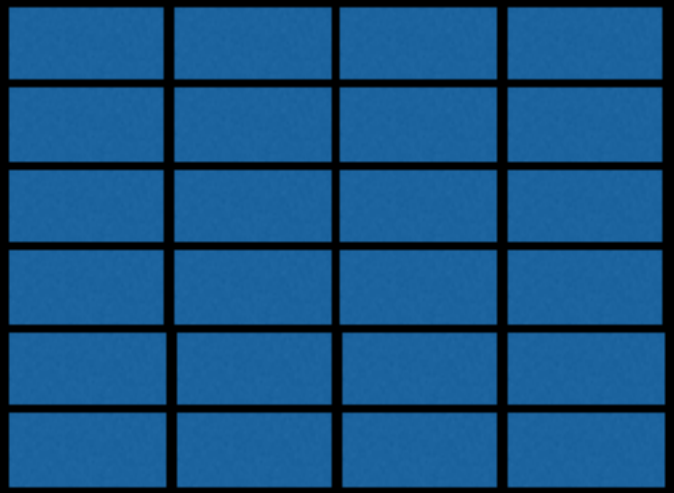

Master Menu
Date: 12-11-1990
Time: 16:21:17
Page 1 of 1
Version 1.00
1. Filing Assistant
Reporting Assistant
Professional Finance Program
EZ Planner
DOS Commands
Managing Your Money
Dr Halo
Word Perfect
Printmaster
↑ or ↓ point to option
ENTER select option
F1 Help
F2 Mail
F9 Menu Maintenance
PgDn Next page
IBM

Client Server Network





Distributed Computing



- Divide work among many machines
- Coordinated central or decentralized
- Examples:
  - Genomics: 100s machines working on a dataset
  - Web Service: 10 machines each taking 1/10th of the web traffic for StackExchange.com
  - Storage: xx,000 machines holding all of Gmail's messages

Distributed computing can do more “work” than the largest single computer.

More storage.
More computing power.
More memory.
More throughput.

Mo' computers, Mo' problems

Thousands of Users

- Bigger risks
- Failures more visible
- Automation mandatory
- Cost containment becomes critical

In response: Radical ideas on

- Reducing risk / Improve safety
- Reliability becomes a competitive differentiator
- New automation paradigms
- Cost and economics

Make peace with failure

Parts are imperfect
Networks are imperfect
Systems are imperfect
Code is imperfect
People are imperfect

Learn how to

FAIL

BETTER


Lucent
5ESS
2000

Buy the best, most reliable computer in the world. It is still going to fail.

If it doesn't, you'll still need to take it down for maintenance.

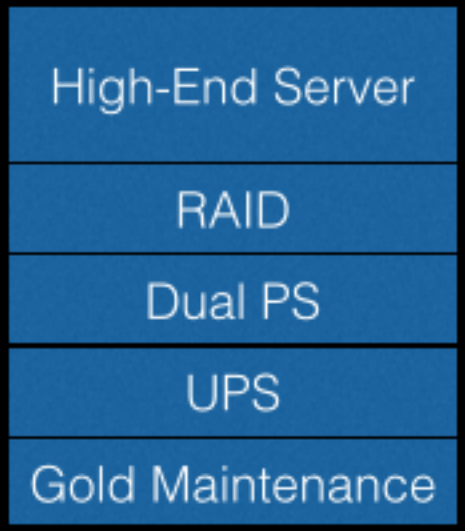
3 ways to fail better

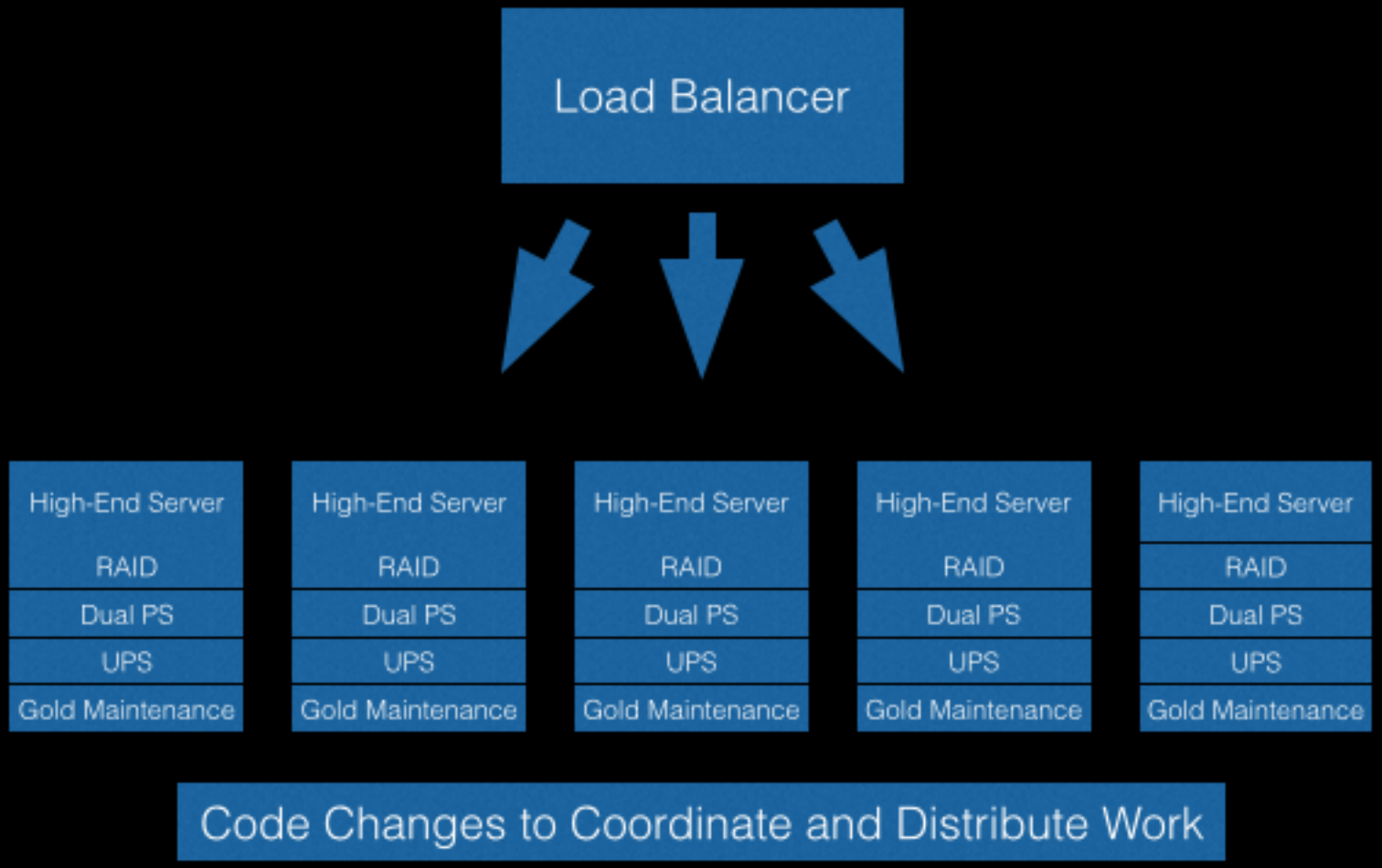
1. Use cheaper, less reliable, hardware.
2. If a process/procedure is risky, do it a lot.
3. Don’t punish people for outages.

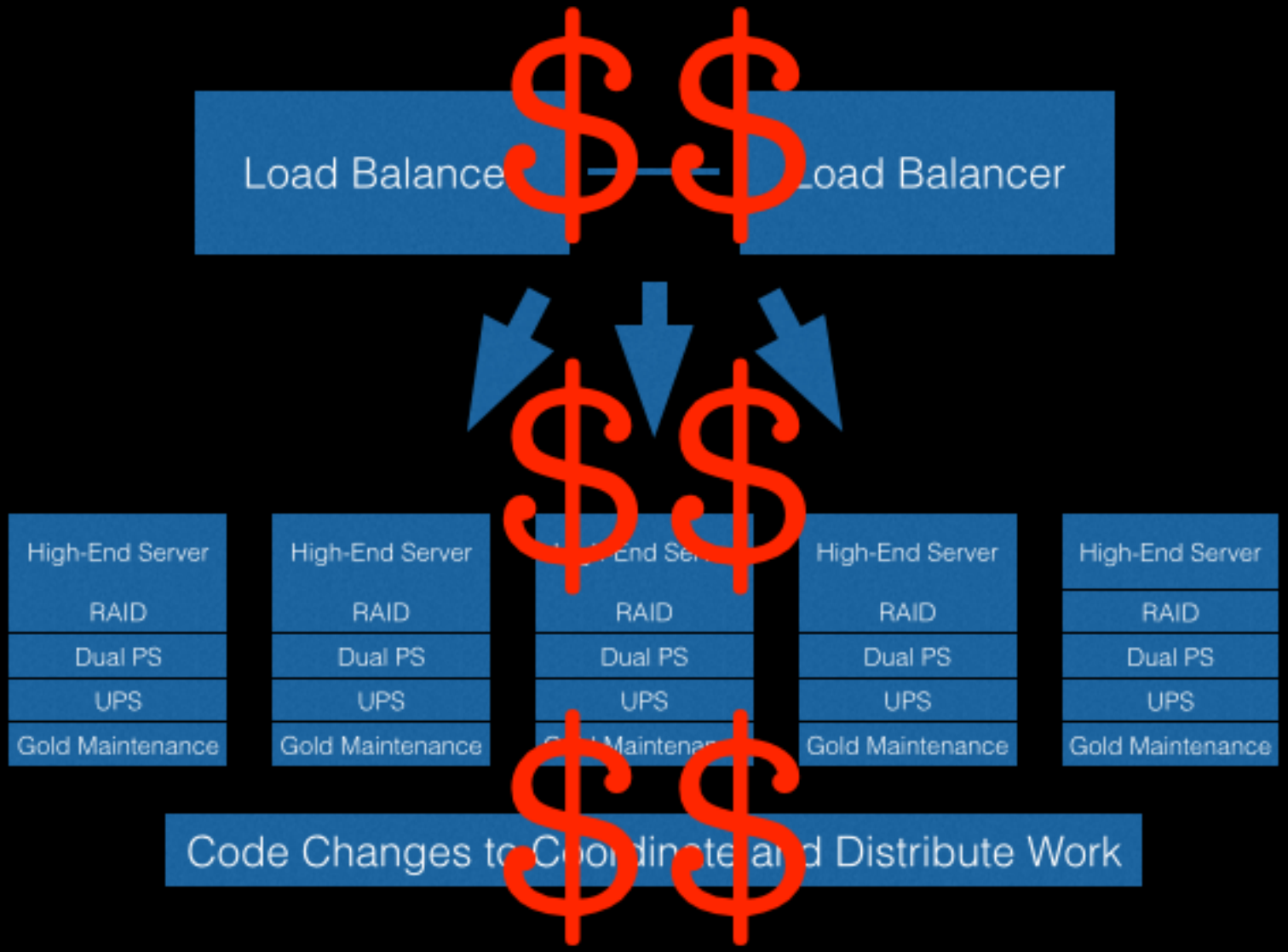
Fail Better Part 1 of 3:

Use cheaper, less reliable, hardware.

- Loss-damage waiver
- Liability
- Personal accident insurance
- Personal effects coverage


High-End Server
RAID
Dual PS
UPS
Gold Maintenance


Load Balancer
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
Code Changes to Coordinate and Distribute Work


Load Balancer
Load Balancer
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
RAID
Dual PS
UPS
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
Code Changes to Coordinate and Distribute Work

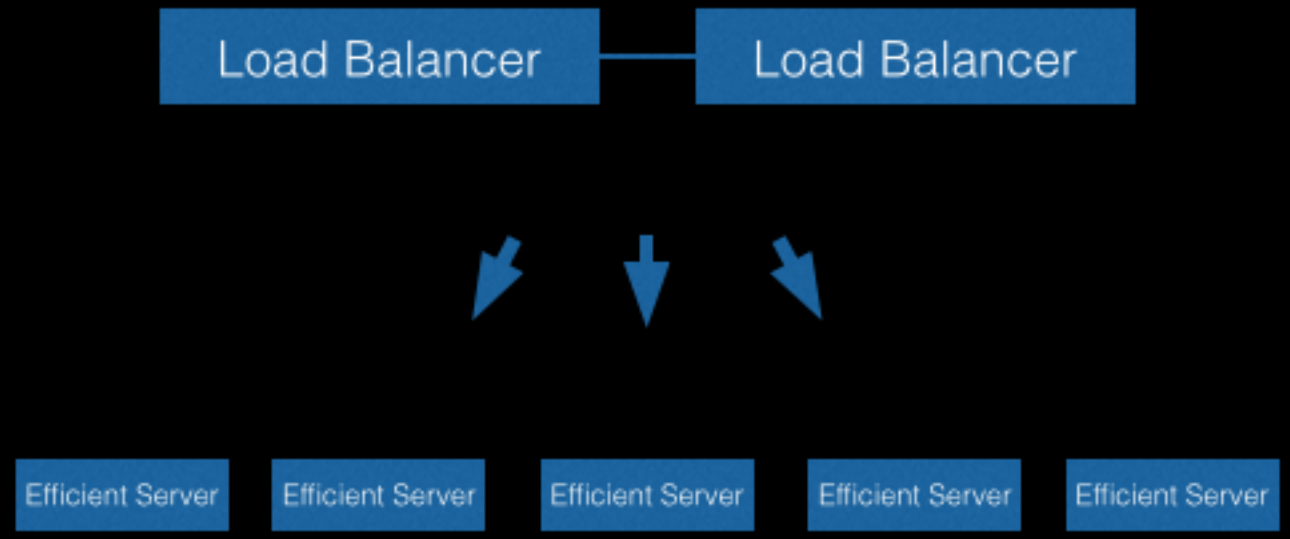
Tip: Prefer Reliability through software, not hardware

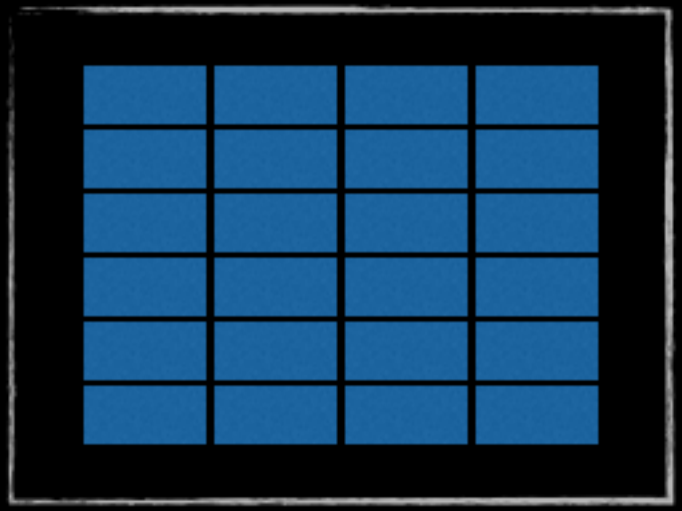
- **Resiliency through software:**
  - Costs to develop. Cheap to deploy.
  - O(1)
- **Resiliency through hardware:**
  - Costs every time you buy a new machine.
  - O(n)

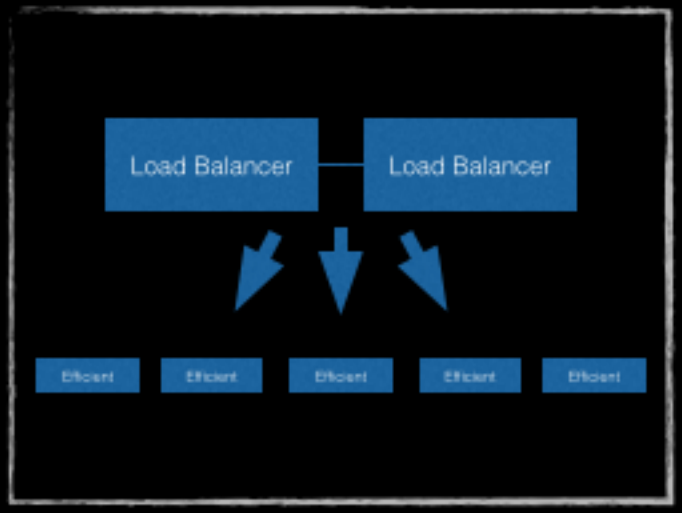
~~Best hardware.~~

Write code so that the system is distributed.

~~Double-spending~~


Load Balancer
Load Balancer
Efficient Server
Efficient Server
Efficient Server
Efficient Server
Efficient Server

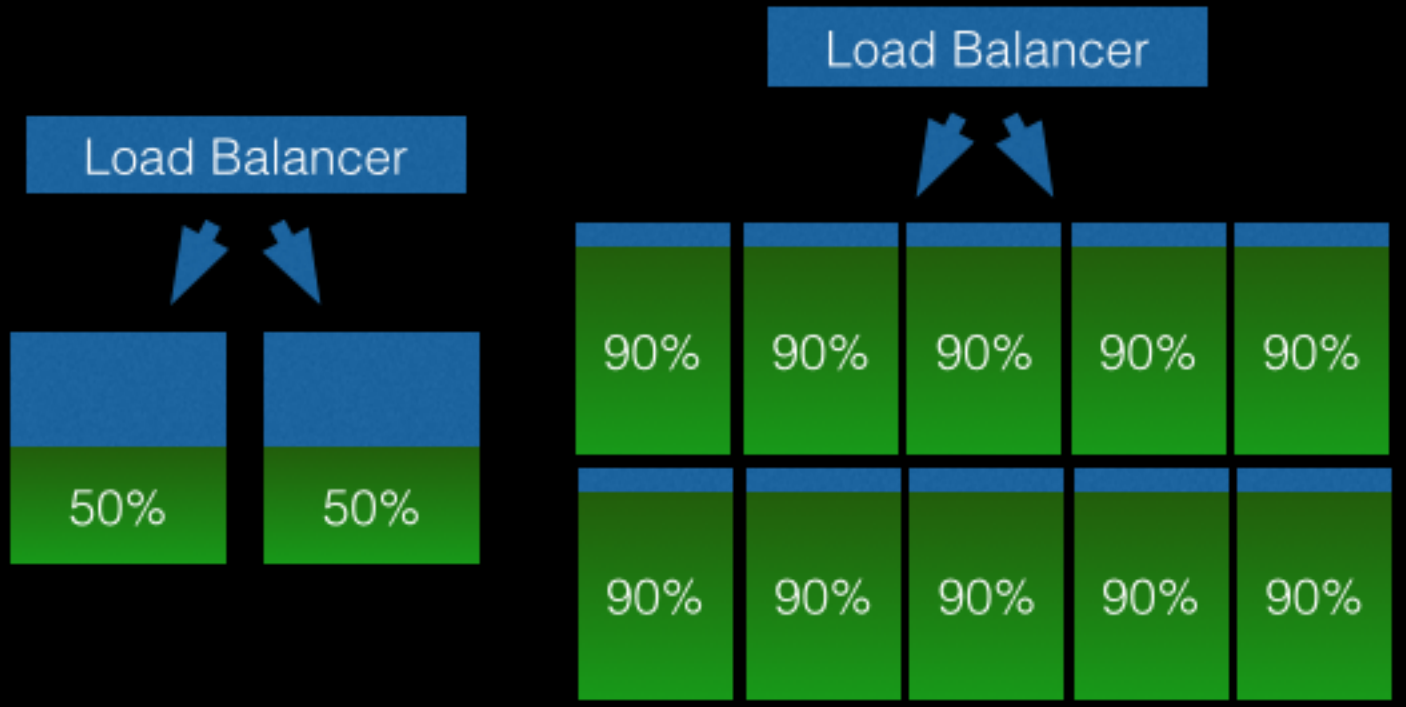




These techniques work for large grids of machines…

…and every-day systems too.

Big resiliency is cheaper



50%
overhead

10%
overhead

Efficiency comes from starting with an SLA and buying enough resiliency to meet it (not exceed it).

Load balancing &
redundancy is just one
way to achieve resiliency.

The cheapest
way to buy
terabytes of RAM.

Fail Better Part 1 of 3:

Use cheaper, less reliable, hardware.

Fail Better Part 2 of 3:

If a process/procedure
is risky, do it a lot.

Risky behavior
vs.
Risky procedures

Risky Behaviors are inherently risky

Smoking

Shooting yourself in the foot

Blindfolded chainsaw juggling

Risky behavior is risky.

Risky Processes can be improved through practice

- Software Upgrades
- Database Failovers
- Network Trunk Failovers
- Hardware Hot Swaps

StackExchange.com Failover from NY or Oregon

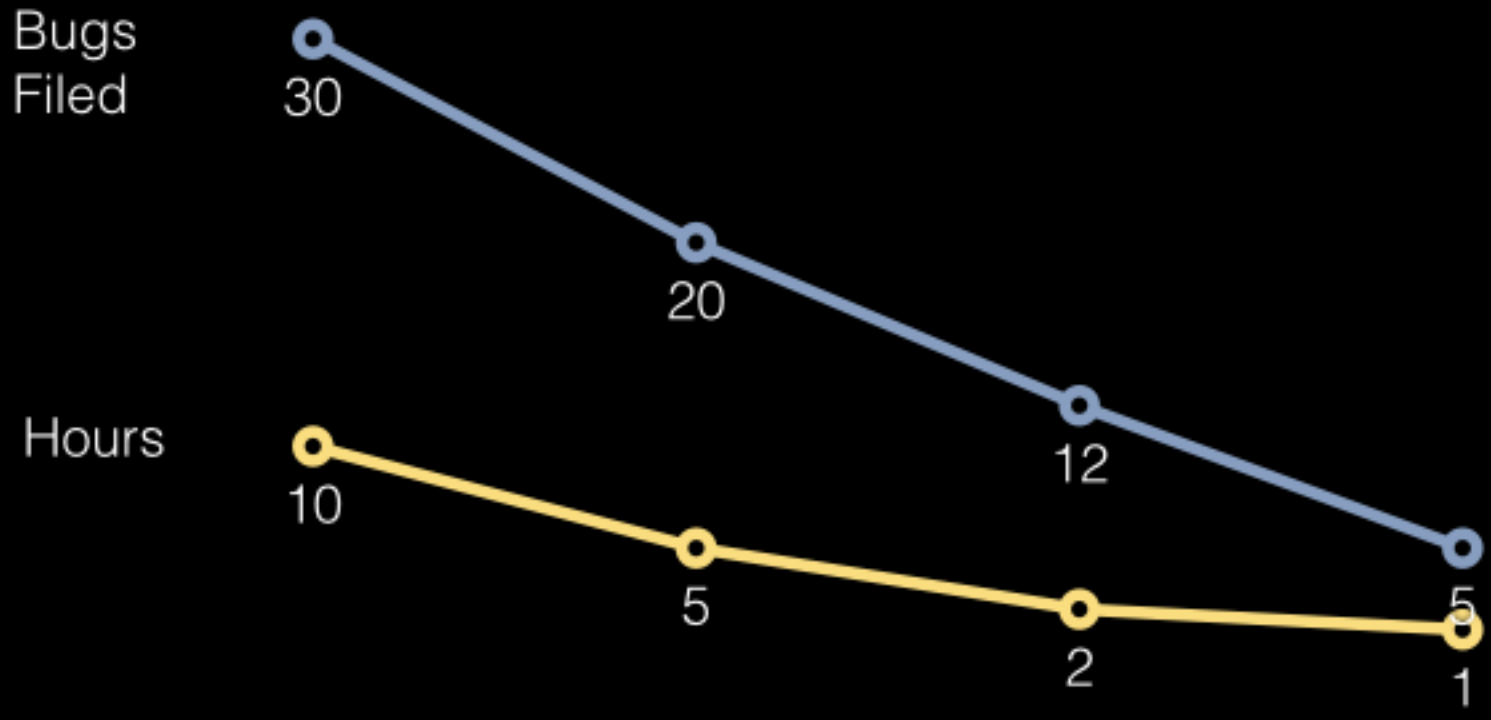
- StackExchange.com has a “DR” site in Oregon.
- StackExchange.com runs on SQL Server with “AlwaysOn” Availability Groups plus…

  Redis, HAproxy, ISC BIND, CloudFlare, IIS, and many home-grown applications

Process was risky

- Took 10+ hours
- Required “hands on” by 3 teams.
- Found 30+ “improvements needed”
- Certain people were S.P.O.F.

Drill Results



Why?

- Each drill “surfaces” areas of improvement.
- Each member of the team gains experience and builds confidence.
- “Smaller Batches” are better

Software Upgrades

- Traditional
  - Months of planning
  - Incompatibility issues
  - Very expensive
  - Very visible mistakes
  - By the time we're done, time to start over again.
- Distributed Computing
  - High frequency (many times a day or week)
  - Fully automated
  - Easy to fix failures
  - Cheap… encourages experiments

“Big Bang” releases
are inherently risky.

Small batches are better

Fewer changes each batch:

- If there are bugs, easier to identify source

Reduced lead time:

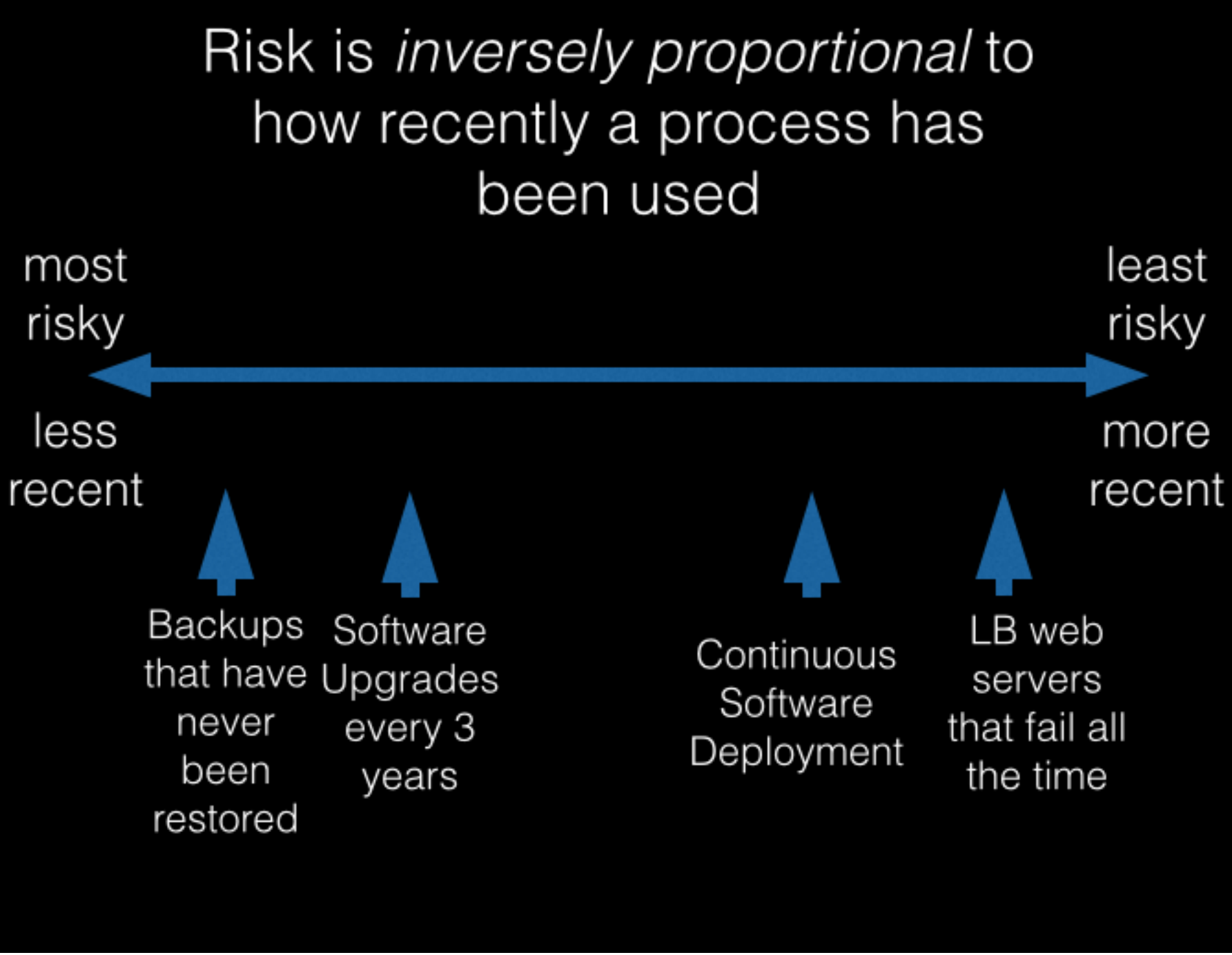
- It is easier to debug code written recently.

Environment has changed less:

- Fewer “external changes” to break on

Happier, more motivated, employees:

- Instant gratification for all involved


Risk is *inversely proportional* to how recently a process has been used
most risky
least risky
less recent
more recent
Backups that have never been restored
Software Upgrades every 3 years
Continuous Software Deployment
LB web servers that fail all the time

Netflix “Chaos Monkey”

- **Randomly reboots machines.**
- **Keeps Netflix “on its toes”.**
- **Part of the Simian Army:**
  - Chaos Monkey (hosts)
  - Chaos Kong (data centers)
  - Latency Monkey (adds random performance delays)

Fail Better Part 2 of 3:

If a process/procedure
is risky, do it a lot.

Fail Better Part 3 of 3:

Don’t punish people for outages.

There will always be outa

Make peace with failure

- Parts are imperfect
- Networks are imperfect
- Systems are imperfect
- People are imperfect

Getting angry about outages is equivalent to expecting them to never happen… which is *irrational*.

Out-dated attitudes about outages

- Expect perfection: 100% uptime
- Punish exceptions:
  - fire someone to “prove we’re serious”
- Results:
  - People hide problems
  - People stop communicating
  - Discourages transparency
  - Small problems get ignored, turn into big problems

New thinking on outages

- Set uptime goals: 99.9% +/- 0.05
- Anticipate outages:
  - Strategic resiliency techniques, oncall system
  - Drills to keep in practice, improve process
- Results:
  - Encourages transparency, communication
  - Small problems addressed, fewer big problems
  - Over-all uptime improved

There are only Contributing Factors

John Allspaw
http://www.kitchensoap.com/2012/02/10/each-necessary-but-only-jointly-sufficient/

After the outage, publish a postmortem document

- People involved write a “blameless postmortem”
  - Identifies what happened, how, what can be done to prevent similar problems in the future.
  - Published widely internally and externally.
- Instead of **blame**, people take **responsibility**:
  - **Responsibility** for implementing long-term fixes.
  - **Responsibility** for educating other teams how to learn from this.

Outage Post-Mortem - 2014-08-25/Outage

Summary:

On Aug 25, 2014 there was an outage of all web properties (Core and Careers) from 3:27pm to 3:32pm NYC-time (approx 7 minutes). The cause was an incorrect change to security settings. The solution was to revert the change via Puppet. Measures being implemented to prevent this problem in the future are listed below.

Outage Type	Sites Down
Outage Timeframe	2014-08-25 19:24, about 7m of downtime
Worst-Case Outage window	7 minutes
Assets affected	All
Summary of causes	Bad change to firewall rules.
Recommendations	Need to refactor firewall rules to be more easy to understand and update; Need to develop better testing methods for firewall rulesets.

Background Information

The intended change: SRE was attempting to update the firewall rules to to permit internal openid calls to work directly rather than going out to the internet and back in.

Outage Schedule of Events

2014-08-25 19:01 da2d38d6a Change pushed to Git
2014-08-25 19:26 Puppet runs on ny-lb05 (pushed bad change / outage BEGINS)
2014-08-25 19:27 Pagerduty and Pingdom page oncall (Tom)
2014-08-25 19:27 @David asked "Who broke everything but chat?" on SRE-team
2014-08-25 19:27 da2d38d6a1 Revert pushed to Git
2014-08-25 19:30 Puppet runs on ny-lb06 (pushed revert)
2014-08-25 19:32 Puppet runs on ny-lb05 (pushes revert) (outage RESOLVED)

Things that went Right

- Use of version control with Puppet means we are able to revert bad changes quickly.
- Everyone worked together to find and fix the problem.

Processes Needing Improvement

- Firewall rules should be refactored to be easier to understand and update.
- Firewall rules need a better testing method.

Immediate to do

- Improve comments in iptables files (there are wrong and misleading comments) (Done: b55e654d9f)

Long term to do

- Move LB firewalls to the new structure being developed.
- Establish better testing methodology for firewall changes.

I dunno about anybody else, but I really like getting these post-mortem reports. Not only is it **nice to know what happened**, but it's also great to see **how you guys handled it** in the moment and **how you plan to prevent these events** going forward. Really neato. Thanks for the great work :)

—-Anna

Fail Better Part 3 of 3:

Don’t punish people for outages.

Take-homes

- **“cloud computing” = “distributed computing”**

1. **Use cheaper, less reliable, hardware**
   - Create reliability through software (when possible)
   - Pay only for the reliability you need
2. **If a process/procedure is risky, do it a lot**
   - Practice makes perfect
   - “Small Batches” improves quality and morale
3. **Don’t punish people for outages**
   - Focus on accountability and take responsibility

We run services, not servers.

- **A “server” is a server even if it is powered off.**
- **A “service” is powered up, running, and useful.**

Services are important because people depend on them.

We run services, not servers.

- **Healthy services run themselves.**
- **We are hired to be awesome in the face of failure.**

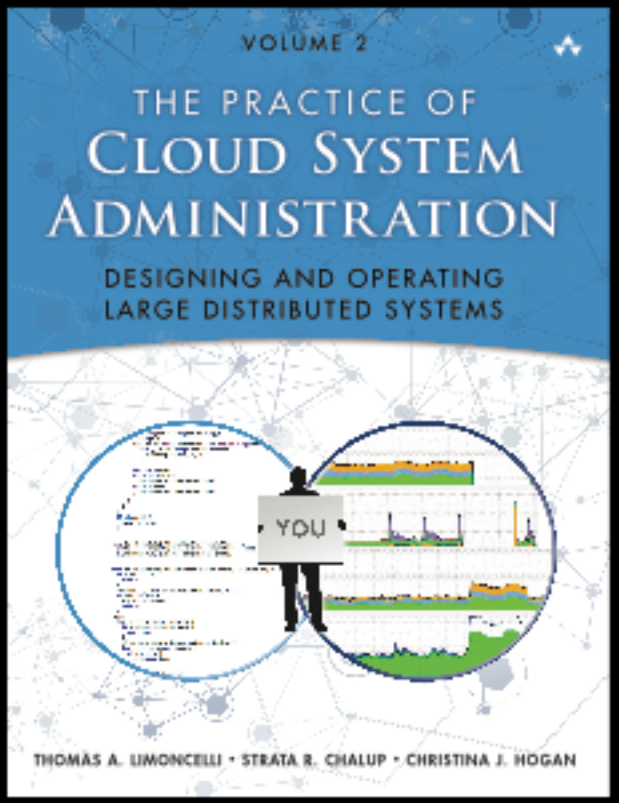
@YesThatTom
http://the-cloud-book.com

Be Awesome

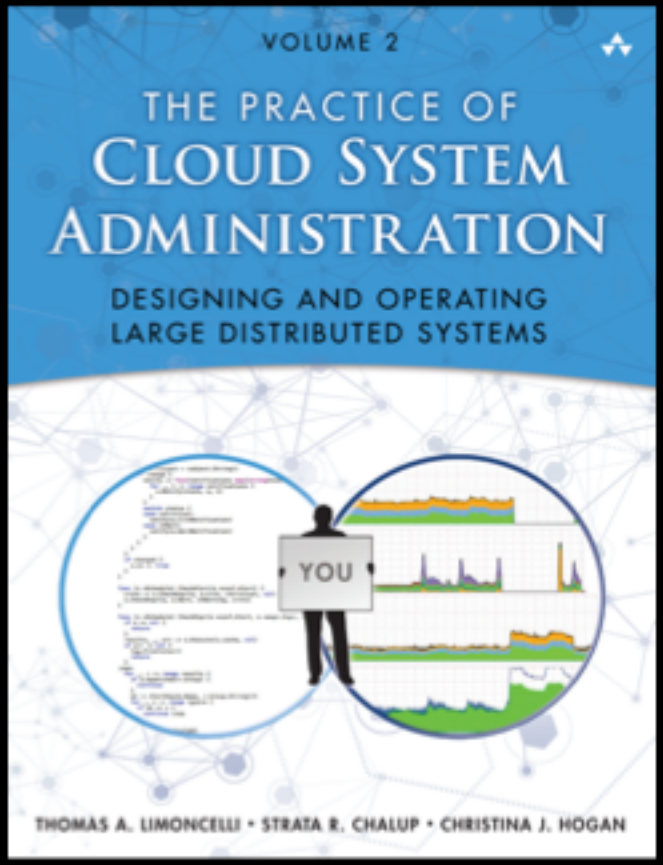
If you liked this talk...

...there's more like it in http://the-cloud-book.com

Save 35%
www.informit.com/TPOSA
Discount code TPOSA35



Radical Ideas from The Practice of Cloud System Administration



Tom Limoncelli, SRE

StackExchange.com

the-cloud-book.com

@YesThatTom

Q&A