CS162
Operating Systems and Systems Programming
Lecture 20

Why Systems Fail and What We Can Do About It

April 9, 2012

Anthony D. Joseph and Ion Stoica
http://inst.eecs.berkeley.edu/~cs162

Goals for Today

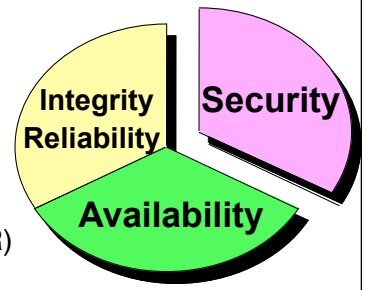
- Definitions for Fault Tolerance
- Causes of system failures
- Possible solutions
  - Single system, Datacenter, geographic diversity

"You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done." —LESLIE LAMPORT

Note: Some slides and/or pictures in the following are adapted from slides from a talk given by Jim Gray at UC Berkeley on November 9, 2000.

Dependability: The 3 ITIES

- **Reliability / Integrity:** does the right thing.
  (Need large MTBF)
- **Availability:** does it now.
  (Need small $\frac{\text{MTTR}}{\text{MTBF+MTTR}}$)

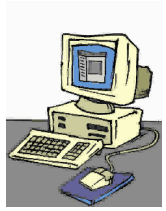


- **System Availability:** if 90% of terminals up & 99% of DB up?
  (=> 89% of transactions are serviced on time)

MTBF or MTTF = Mean Time Between (To) Failure
MTTR = Mean Time To Repair

Fault Tolerance versus Disaster Tolerance

- **Fault-Tolerance:** mask local faults
  - Redundant HW or SW
  - RAID disks
  - Uninterruptible Power Supplies
  - Cluster Failover



- **Disaster Tolerance:** masks site failures
  - Protects against fire, flood, sabotage,..
  - Redundant system and service at remote site.
  - Use design diversity

High Availability System Classes

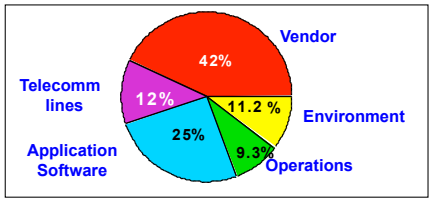
Availability %	Downtime per year	Downtime per month	Downtime per week
90% (“one nine”)			
99% (“two nines”)			
99.9% (“three nines”)			
99.99% (“four nines”)			
99.999% (“five nines”)			
99.9999% (“six nines”)			

GOAL: Class 6

UnAvailability ~ MTTR/MTBF
can cut it in ½ by cutting MTTR *or* MTBF

Case Study – Japan

“Survey on Computer Security”, Japan Info Dev Corp., March 1986. (trans: Eiichi Watanabe).

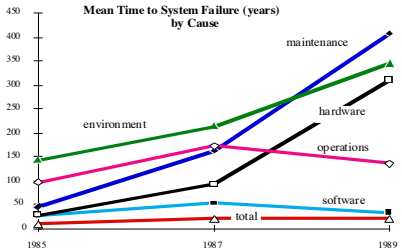


Vendor (hardware and software)	5 Months
Application software	9 Months
Communications lines	1.5 Years
Operations	2 Years
Environment	2 Years
	10 Weeks

1,383 institutions reported (6/84 - 7/85)
7,517 outages, MTBF ~ 10 weeks, avg duration ~ 90 MINUTES

To Get 10 Year MTBF, Must Attack All These Areas

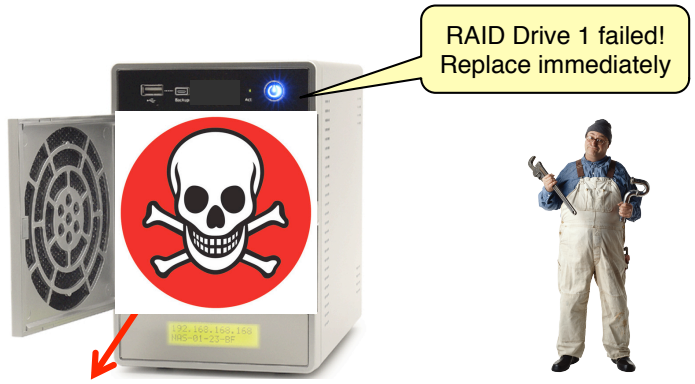
Case Studies - Tandem Trends
Reported MTBF by Component



	1985	1987	1990	
SOFTWARE	2	53	33	Years
HARDWARE	29	91	310	Years
MAINTENANCE	45	162	409	Years
OPERATIONS	99	171	136	Years
ENVIRONMENT	142	214	346	Years
SYSTEM	**8**	**20**	**21**	**Years**

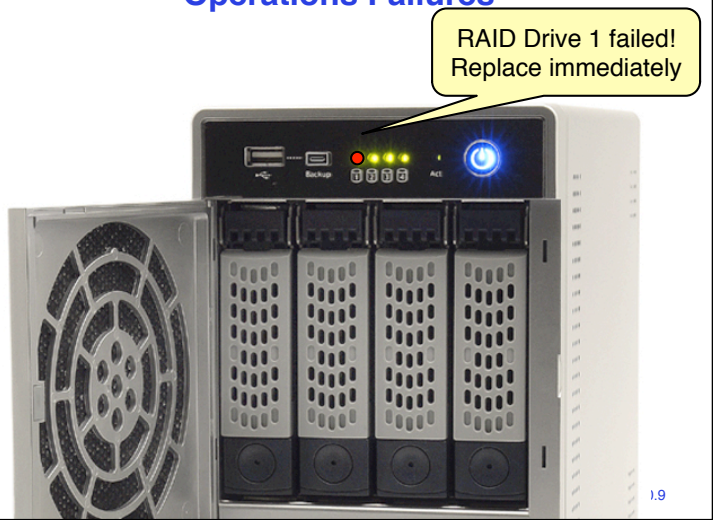
Problem: Systematic Under-reporting

Operations Failures



What went wrong??

Operations Failures



Causal Factors for Unavailability

Lack of best practices:

- Change control
- Monitoring of the relevant components
- Requirements and procurement
- Operations
- Avoidance of network failures, internal application failures, and external services that fail
- Physical environment, and network redundancy
- Technical solution of backup, and process solution of backup
- Physical location, infrastructure redundancy
- Storage architecture redundancy

Ulrik Franke et al: Availability of enterprise IT systems - an expert-based Bayesian model

Cloud Computing Outages 2011

Vendor	When	Duration	What Happened & Why
Apple iPhone 4S Siri	November 2011	1 Day	Siri loses even the most basic functionality when Apples servers are down. Because Siri depends on servers to do the heavy computing required for voice recognition, the service is useless without that connection. Network outages caused the disruption according to Apple.
Blackberry outage	October 2011	3 Days	Outage was caused by a hardware failure (core switch failure) that prompted a "ripple effect" in RIM's systems. Users in Europe, Middle East, Africa, India, Brazil, China and Argentina initially experienced email and message delays and complete outages and later the outages spread to North America too. Main problem is message backlogs and the downtime produced a huge queue of undelivered messages causing delays and traffic jams.
Google Docs	September 2011	1 Hour	Google Docs word collaboration application cramp, shutting out millions of users from their document lists, documents, drawings and Apps Scripts. Outage was caused by a memory management bug software engineers triggered in a change designed to "improve real time collaboration within the document list.
Windows Live services - Hotmail & SkyDrive	September 2011	2 Hours	Users did not have any data loss during the outage and the interruption was due to an issue in Domain Name Service (DNS). Network traffic balancing tool had an update and the update did not work properly which caused the issue.
Amazon's EC2 cloud &	August 2011	1-2 days	Transformer exploded and caught fire near datacenter that resulted in power outage due to generator failure. Power back up systems at both the data centers failed causing power outages. Transformer explosion was caused by lightening strike but disputed by local utility provider.
Microsoft's BPOS	August 2011	1-2 days	Transformer exploded and caught fire near datacenter that resulted in power outage due to generator failure. Power back up systems at both the data centers failed causing power outages. Transformer explosion was caused by lightening strike but disputed by local utility provider.

From: http://analysiscasestudy.blogspot.com/

Cloud Computing Outages 2011

Vendor	When	Duration	What Happened & Why
Amazon Web Services	April, 2011	4 Days	During the upgrade, the traffic shift was executed incorrectly and rather than routing the traffic to the other router on the primary network, the traffic was routed onto the lower capacity redundant EBS network. This led to Amazon Elastic Block Store ("EBS") volumes in a single Availability Zone within the US East Region that became unable to service read and write operations. It also impacted the Relational Database Service ("RDS"). RDS depends upon EBS for database and log storage, and as a result a portion of the RDS databases hosted in the primary affected Availability Zone became inaccessible.
Microsoft BPOS Outages	May 2011	2 Hours	Paying customers' email was delayed by as much as nine hours. Delay as outgoing messages started getting stuck in the pipeline.
Twitter Outages	March & Feb 2011	1-4 Hours	Outages due to over capacity and moving operations to new data center.
Intuit Quick Books Online	March 2011	2 Days	Service failures on human error during scheduled maintenance operations. Inuit changed its network configuration and inadvertently blocked customer access to a portion of the company's servers. A surge in traffic overloaded the servers when connectivity was restored, so the company opted to restore service.
Google Mail and Apps Outage	February 2011	2 Days	Google mail and Google Apps users experienced login errors and empty mailboxes. Google Engineering determined that the root cause was a bug inadvertently introduced in a Gmail storage software update. The bug caused the affected users' messages and account settings to become temporarily unavailable from the datacenters.

From: http://analysiscasestudy.blogspot.com/

Cloud Computing Outages 2010

Vendor	When	Duration	What Happened & Why
Hotmail Outage	December 2010	3 Days	A number of our users reported their email messages and folders were missing from their Hotmail accounts. Error occurred from a script that was meant to delete dummy accounts created for automated testing and inadvertently targeted 17,000 real accounts instead.
Skype Outage	December 2010	1 Day	Cluster of support servers responsible for offline instant messaging became overloaded and the P2P network became unstable and suffered a critical failure. A supernode is important to the P2P network acting like a directory, supporting other Skype clients, helping to establish connections between them etc. The failure of 25–30% of supernodes in the P2P network resulted in an increased load on the remaining supernodes.
Paypal Outage	November 2010	3 Hours	A network hardware failure was the trigger for an outage. The hardware failure was worsened by problems in shifting traffic to another data center, resulting in about 90 minutes of downtime.
Facebook Outage	September 2010	2 ½ Hours	Outage due to an error condition. An automated system for verifying configuration values ended up causing much more damage than it fixed. Every single client saw the invalid value and attempted to fix it that led to a query to a database cluster and cluster was overloaded with thousand of queries per second. Even after fixing problem stream of queries continued.
Microsoft BPOS Outages	August & September 2010	2 Hours	A design issue in the upgrade that caused unexpected impact, but the issue resulted in a 2 hour period of intermittent access for BPOS organizations served from North America.
Wikipedia Outage	July & March 2010	2-3 Hours	In July, the power failure is understood to have affected Wikimedia's '*pmtpa*' cluster. Due to the temporary unavailability of several critical systems and the large impact on the available systems capacity, all Wikimedia projects went down. In March, Wikimedia servers overheated in the organization's European data center and shut themselves off automatically. Wikimedia then switched all its traffic to its server cluster in Florida, but the failover process, which involves changing servers' DNS entries, malfunctioned, knocking the organization's sites offline around the world.
Hosting.com Outage	June 2010	2 Hours	Failure of a Cisco switch at the Newark, N.J., data center caused intermittent network connectivity. Dedicated switch had failed, the second failover switch had crashed as well and the problem was caused by a software bug.
Twitter.com outage	June 2010	5 hours	Increased activity on the site, combined with system enhancements and upgrades, have uncovered networking issues. Incidences of poor site performance and a high number of errors due to one of the internal sub-networks being over-capacity.

Cloud Computing Outages 2009

Vendor	When	Duration	What Happened & Why
Salesforce.com Outage	January 2010, 2009	1-2 Hours	Outages were caused by server disruption, when a core network device failed, stopping all data from being processed in Japan, Europe, and North America. The technical reason for the outage: a core network device had failed, due to memory allocation errors. The backup plan, which was supposed to trigger a cut-over to a redundant system, also failed.
Amazon's EC2	June 2009	4-5 Hours	A lightning storm caused damage to a single Power Distribution Unit (PDU) in a single Availability Zone
eBay Paypal	August 2009	1-4 Hours	Online payments system failed a couple of times led to non completion of transactions. Network hardware issue is blamed for outage.
Twitter	August 2009	½ Day	A denial-of-service attack was blamed for the problem
Google Gmail	September 2009	2 hours 2 times	Reasons from vendors include routing errors to server maintenance issues.
Microsoft Sidekick	October 2009	6 days	Microsoft's Danger server farm, that holds the cloud T-Mobile Sidekick subscriber's data crashed, depriving users of their calendar, address book, and other key data. Critical data was lost during outage.
Rackspace.com Outage	June 2009 December 2009	1 Day 1 Hour	Power outage and subsequent power generator failures that caused servers to fail. Company was forced to pay out between $2.5 million and $3.5 million in service credits to customers. The issues resulted from a problem with a router used for peering and backbone connectivity located outside the data center at a peering facility, which handles approximately 20% of Rackspace's Dallas traffic. The router configuration error was part of final testing for data center integration between the Chicago and Dallas facilities.

From: http://analysiscasestudy.blogspot.com/

Fault Model

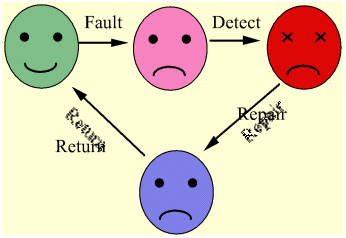
- Failures are independent*
  So, single fault tolerance is a big win
- Hardware fails fast (blue-screen, panic, …)
- Software fails-fast (or stops responding/hangs)
- Software often repaired by reboot:
  - Heisenbugs – Works On Retry
  - (Bohrbugs – Faults Again On Retry)
- Operations tasks: major source of outage
  - Utility operations – UPS/generator maintenance
  - Software upgrades, configuration changes

Some Fault Tolerance Techniques

- Fail fast modules: work or stop
- Spare modules: yield instant repair time
- Process/Server pairs: Mask HW and SW faults
- Transactions: yields ACID semantics (simple fault model)

Fail-Fast is Good, Repair is Needed

Lifecycle of a module
fail-fast gives
short fault latency



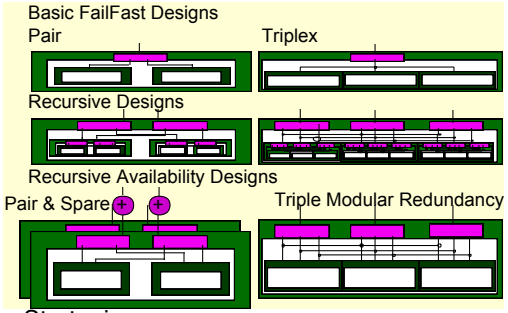
High Availability
is low UN-Availability

Unavailability ~ $\frac{\text{MTTR}}{\text{MTBF}}$

Improving either MTTR or MTBF gives benefit

Simple redundancy does not help much

Hardware Reliability/Availability
(how to make HW fail fast)

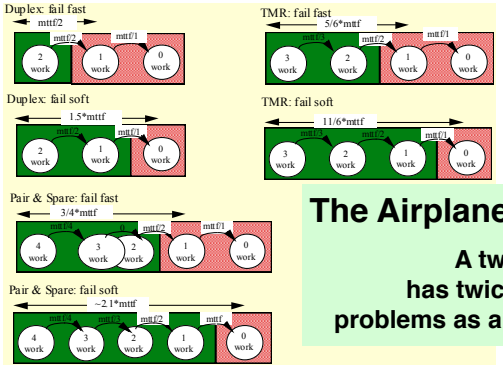


Comparitor Strategies: (in recursive pairs, parent knows which is bad)

Duplex: Fail-Fast: fail if either fails (e.g. duplexed CPUs)
vs Fail-Soft: fail if both fail (e.g. disc, network,...)

Triplex: Fail-Fast: fail if 2 fail (triplexed cpus)
Fail-Soft: fail if 3 fail (triplexed FailFast CPUs)

Redundant Designs have Worse MTBF!



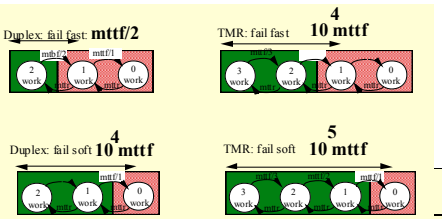
The Airplane Rule:

A two-engine airplane has twice as many engine problems as a one engine plane

THIS IS NOT GOOD: Variance is lower but MTBF is worse

Simple redundancy does not improve MTBF (sometimes hurts)

Add Repair: Get 10^4 Improvement



Availability estimates
1 year MTTF modules
12-hour MTTR

	MTTF	EQUATION	COST
SIMPLEX	1 year	MTTF	1
DUPLEX: FAIL FAST	~0.5 years	- MTTF/2	2+ε
DUPLEX: FAIL SOFT	~1.5 years	- MTTF(3/2)	2+ε
TRIPLEX: FAIL FAST	.8 year	- MTTF(5/6)	3+ε
TRIPLEX: FAIL SOFT	1.8 year	- 1.8MTTF	3+ε
Pair and spare: FAIL-FAST	~.7 year	- MTTF(3/4)	4+ε
TRIPLEX WITH REPAIR	>10^5 years	$MTTF^3/3MTTR^2$	3+ε
Duplex fail soft + REPAIR	>104 years	$MTTF^2/2MTTR$	4+ε

Software Techniques: Learning from Hardware

- Recall that most outages are not hardware
- Most outages in Fault Tolerant Systems are SOFTWARE
- Fault avoidance techniques: Good & Correct design
- After that: Software Fault Tolerance Techniques:

  Modularity (isolation, fault containment)

  N-Version Programming: N-different implementations

  Programming for Failures: Programming paradigms that assume failures are common and hide them

  Defensive Programming: Check parameters and data

  Auditors: Check data structures in background

  Transactions: to clean up state after a failure

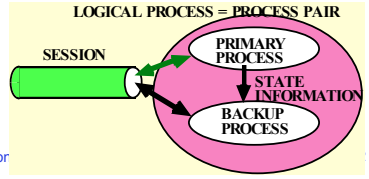
Paradox: Need Fail-Fast Software

Fail-Fast and High-Availability Execution

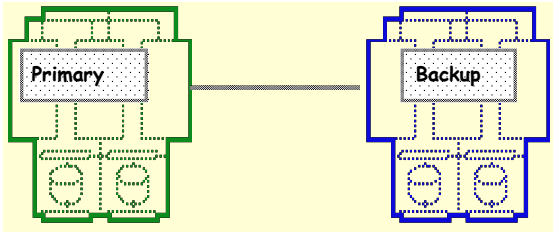
Process Pairs: Instant restart (repair)

Use Defensive programming to make a process fail-fast
Have restarted process ready in separate environment
Second process "takes over" if primary faults

- If software fault (bug) is a Bohrbug, then *there is no repair*
  "wait for the next release" or "get an emergency bug fix" or "get a new vendor"
- If software fault is a Heisenbug, then repair is
  "reboot and retry" or "switch to backup process (instant restart)"
- Tolerates HW faults too!
- Millisecond repair times



Server System Pairs for High Availability



- Programs, Data, Processes Replicated at 2+ sites
  - Pair looks like a single system
- System becomes logical concept
  - Like Process Pairs: System Pairs.
- Backup receives transaction log (spooled if backup down)
- If primary fails or operator switches, backup offers service

Administrivia

- Project 3 code due Thu 4/12 at 11:59pm
- Final exam Fri 5/11 11:30-2:30pm in 145 Dwinelle
  - Comprehensive, closed book/notes
  - Two double-sided handwritten pages of notes allowed
- PSA: Backups!
  - World Backup Day was March 31st
  - Perform regular local backups
  - Test recovering a file
  - File vs disaster recovery
  - Offsite options: box.net, Dropbox (http://db.tt/k3A3n3T)

5min Break

Datacenter is new "server"

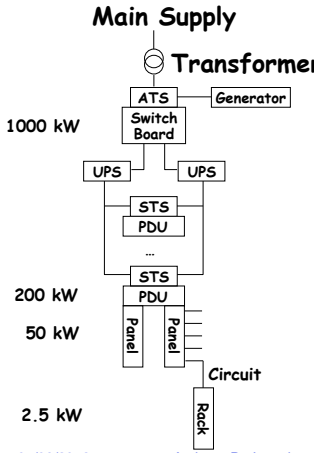
- *"Program"* == Web search, email, map/GIS, …
- *"Computer"* == 1000's computers, storage, network
- Warehouse-sized facilities and workloads
- New datacenter ideas (2007-2008): truck container (Sun), floating (Google), datacenter-in-a-tent (Microsoft)

photos: Sun Microsystems & datacenterknowledge.com

Datacenter Architectures

- Major engineering design challenges in building datacenters
  - One of Google's biggest secrets and challenges
  - Facebook creating open solution: http://www.facebook.com/note.php?note_id=10150148003778920
  - Very hard to get everything correct!
- Example: AT&T Miami, Florida Tier 1 datacenter
  - Minimum N+1 redundancy factor on all critical infrastructure systems (power, comms, cooling, …)
  - Redundant communications: dual uplinks to AT&T global backbone
  - What about power?

Typical Tier-2 One Megawatt Datacenter



- Reliable power: Mains + Generator, Dual UPS
- Units of Aggregation
  - Rack (10-80 nodes) → PDU (20-60 racks) → Facility/ Datacenter
- Power is 40% of DC costs
- Over 20% of entire DC costs is in power redundancy

X. Fan, W-D Weber, L. Barroso, "Power Provisioning for a Warehouse-sized Computer," ISCA'07, San Diego, (June 2007).

AT&T Internet Data Center Power

Commercial Power Feeds

2 commercial feeds, Each at 13,800VAC
Located near Substation supplied from 2 different grids

AT&T Internet Data Center Power

- Paralleling switch gear
- Automatically powers up all generators when Commercial power is interrupted for more than 7 seconds
  - Generators are shed to cover load as needed
  - Typical transition takes less than 60 seconds
- Manual override available to ensure continuity if automatic start-up should fail

Emergency Power Switchover

AT&T Internet Data Center Power

Back-up Power – Generators and Diesel Fuel

- Four (4) 2,500 kw Diesel Generators Providing Standby Power, capable of producing 10 MW of power
- Two (2) 33,000 Gallon Aboveground Diesel Fuel Storage Tanks

AT&T Internet Data Center Power

- UPS consisting of four battery strings
- Battery strings contain flooded cell Lead Acid batteries
- A minimum of 15 minutes of battery backup available at Full load
- Remote status monitoring of battery strings

UPS Batteries

AT&T Internet Data Center Power

- Motor-Generator pairs clean power
- Eliminate spikes, sags, surges, transients, and all other Over/Under voltage And frequency conditions

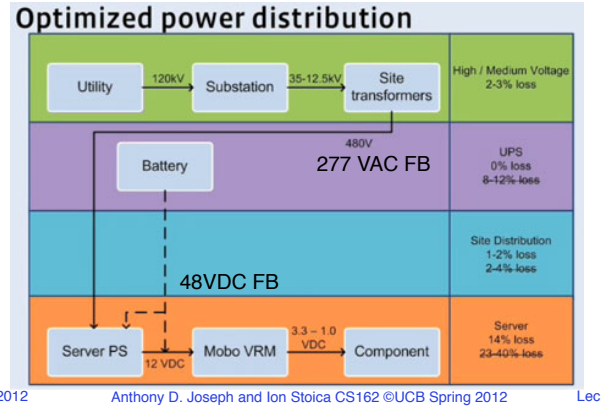
Uninterruptible Power Supply (UPS)

- Four UPS Modules connected in a Ring Bus configuration
- Each Module rated at 1000kVA
- Rotary Type UPS by Piller

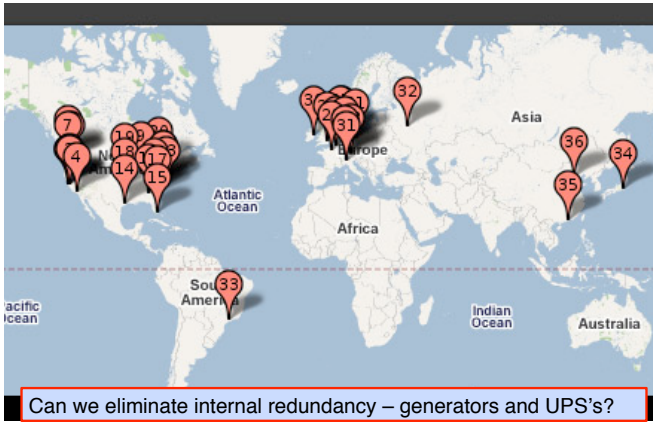
AT&T Enterprise Hosting Services briefing 10/29/2008

Alternate Power Distribution Model

- Google and Facebook replaced central UPS with individual batteries*



Add Geographic Diversity to *Reduce* Single Points of Failure*

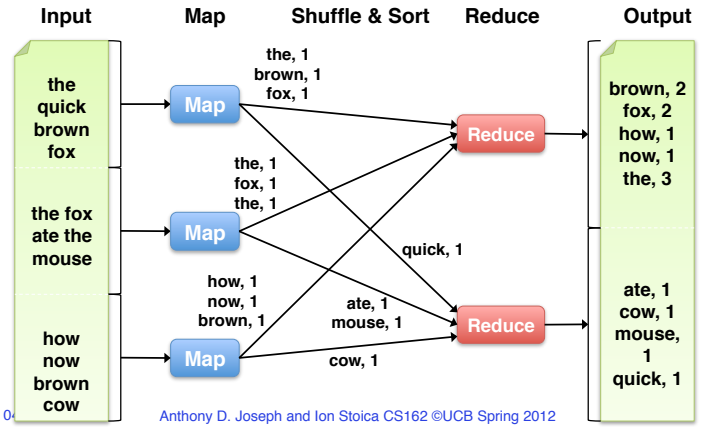


Can we eliminate internal redundancy – generators and UPS's?

Software Fault Tolerance: MapReduce

- First widely popular programming model for data-intensive apps on commodity clusters
- Published by Google in 2004
  - Processes 20 PB of data / day
- Popularized by open-source Hadoop project
  - 40,000 nodes at Yahoo!, 70 PB at Facebook
- Programming model
  - Data type: key-value *records*
    - » Map function: (K_{in}, V_{in}) ➔ list(K_{inter}, V_{inter})
    - » Reduce function: $(K_{inter}, \text{list}(V_{inter}))$ ➔ list(K_{out}, V_{out})

Word Count Execution



Fault Tolerance in MapReduce

1. If a task crashes:
   - Retry on another node
     - » OK for a map because it had no dependencies
     - » OK for reduce because map outputs are on disk
   - If the same task repeatedly fails, fail the job

➢ **Note: For the fault tolerance to work, tasks must be *deterministic* and *side-effect-free***

Fault Tolerance in MapReduce

2. If a node crashes:
   - Relaunch its current tasks on other nodes
   - Relaunch any maps the node previously ran
     - » Necessary because their output files are lost

Fault Tolerance in MapReduce

3. If a task is going slowly (straggler):
   - Launch second copy of task on another node
   - Take output of whichever copy finishes first

- Critical for performance in large clusters

Takeaways

- By providing a data-parallel programming model, MapReduce can control job execution in useful ways:
  - Automatic division of job into tasks
  - Placement of computation near data
  - Load balancing
  - Recovery from failures & stragglers

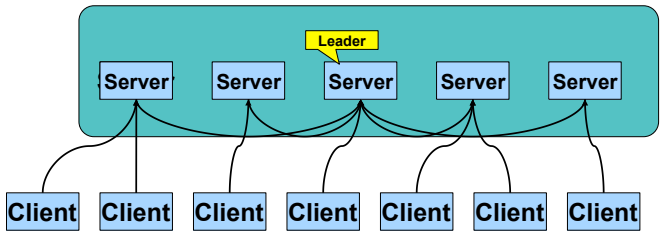
Issues with MapReduce

- Hard to express more complex programs
  - E.g. word count + a sort to find the top words
  - Need to write many different map and reduce functions that are split up all over the program
  - Must write complex operators (e.g. join) by hand
- Acyclic data flow -> poor support for applications that need to *reuse* pieces of data
  - Iterative algorithms (e.g. machine learning, graphs)
  - Interactive data mining (e.g. Matlab, Python, SQL)
- Alternative: Spark programming paradigm

Apache ZooKeeper

- Highly available, scalable, distributed coordination kernel
  - Leader Election, Group Membership, Work Queues, Sharding
  - Event Notifications/workflow, Config, and Cluster Mgmt
- Provides:
  - File API without partial reads/writes and no renames
  - Ordered updates and strong persistence guarantees
  - Conditional updates (version), Watches for data changes
- API:
  - String create(path, data, acl, flags)
  - void delete(path, expectedVersion)
  - Stat setData(path, data, expectedVersion)
  - (data, Stat) getData(path, watch)
  - Stat exists(path, watch)
  - String[] getChildren(path, watch)

ZooKeeper Service



- All servers store a copy of the data (in memory)
- A leader is elected at startup, or upon current leader failure
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

Summary

- Focus on Reliability/Integrity and Availability
  - Also, Security (see next two lectures)
- Use HW/SW Fault-Tolerance techniques to increase MTBF and reduce MTTR
  - Assume the unlikely is likely
- Make operations bulletproof: configuration changes, upgrades, new feature deployment, …
- Apply replication at all levels (including globally)
- Leverage software to build reliable systems from unreliable components