Recovery Oriented Computing (ROC)

Dave Patterson and a cast of 1000s:

Aaron Brown, Pete Broadwell, George Candea[†], Mike Chen, James Cutler[†], Prof. Armando Fox[†], Emre Kıcıman[†], David Oppenheimer, and Jonathan Traupman

U.C. Berkeley, [†]Stanford University

November 2002

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
- **The future:** how will history judge us?
- **Alternative future:** Recovery-Oriented Computing
- **ROC vs. Traditional Fault Tolerance**
- **ROC principles and quick examples**

The past: research goals and assumptions of last 20 years

- **Goal #1: Improve performance**
- **Goal #2: Improve performance**
- **Goal #3: Improve cost-performance**
- **Simplifying Assumptions**
  - Humans are perfect (they don't make mistakes during installation, wiring, upgrade, maintenance or repair)
  - Software will eventually be bug free
    (Hire better programmers!)
  - Hardware MTBF is already very large (~100 years between failures), and will continue to increase
  - Maintenance costs irrelevant vs. Purchase price (maintenance a function of price, so cheaper helps)

2000 Downtime Costs (per Hour)

Brokerage operations	$6,450,000
Credit card authorization	$2,600,000
Ebay (1 outage 22 hours)	$225,000
Amazon.com	$180,000
Package shipping services	$150,000
Home shopping channel	$113,000
Catalog sales center	$90,000
Airline reservation center	$89,000
Cellular service activation	$41,000
On-line network fees	$25,000
ATM service fees	$14,000

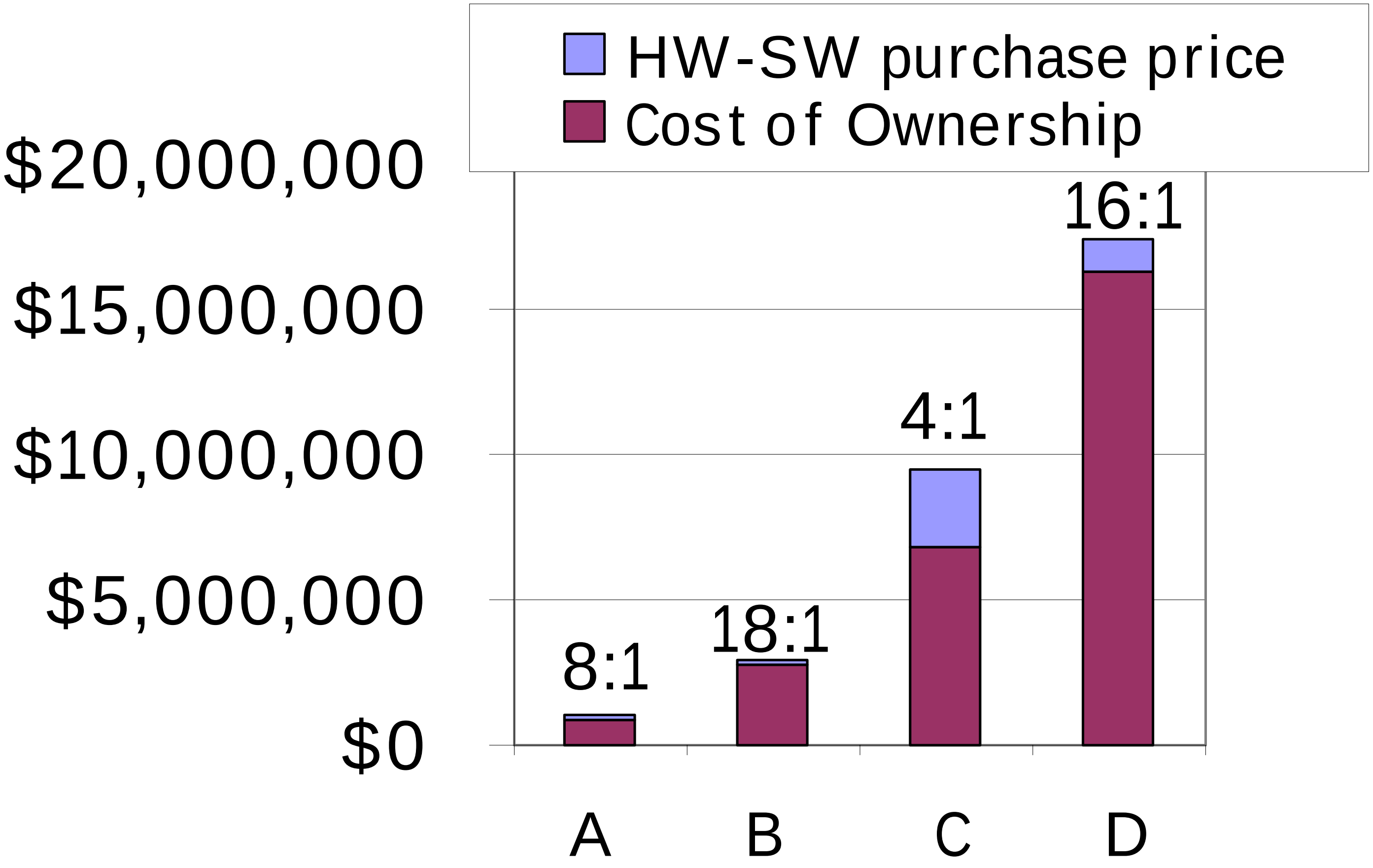
Sources: InternetWeek 4/3/2000 + *Fibre Channel: A Comprehensive Introduction*, R. Kembel 2000, p.8. "...based on a survey done by Contingency Planning Research."

Lost Productivity Ups Outage Cost

- **Amazon 2001: Revenue $3.1B, 7744 employees**
- **Revenue (24x7): $350k per hour**
- **Employee productivity costs: $250k per hour**
  - Assuming average annual salary and benefits is $85,000 and 50 working hours week
- **Total Downtime Costs: $600,000 per hour**
- **Note: Employee cost/hour comparable to revenue, even for an Internet company**

Source: D. Patterson A simple way to estimate the cost of downtime. 16th Systems Administration Conference, November 2002.

Total Cost of Ownership: Ownership vs. Purchase

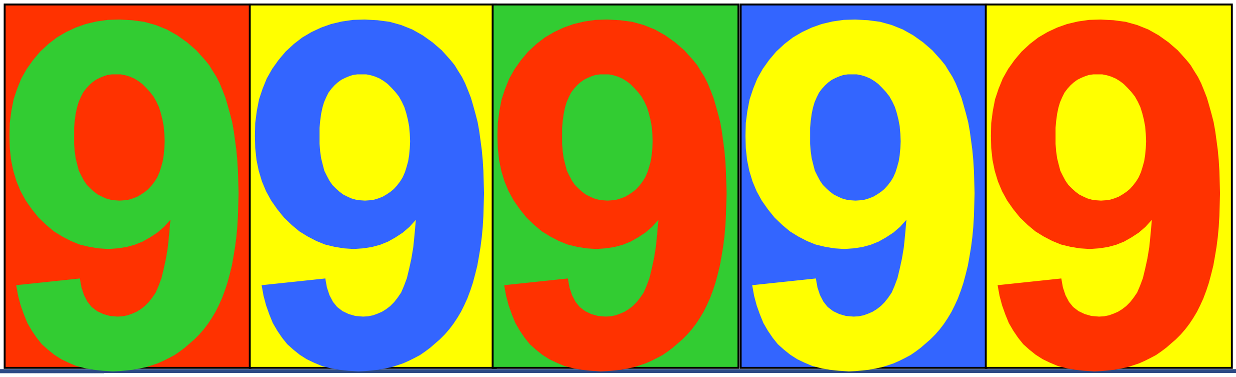


- **HW/SW decrease vs. Salary Increase**
  - 142 sites, 1200-7600 users/site, $2B/yr sales

Source: "The Role of Linux in Reducing the Cost of Enterprise Computing“, IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com

Dependability: Claims of 5 9s?

- 99.999% availability from telephone company?
  - **AT&T switches < 2 hours of failure in 40 years**
- Cisco, HP, Microsoft, Sun … claim 99.999% availability claims (5 minutes down / year) in marketing/advertising
  - **HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee "in certain pre-defined, pre-tested customer environments"**
  - **Environmental? Application? Operator?**



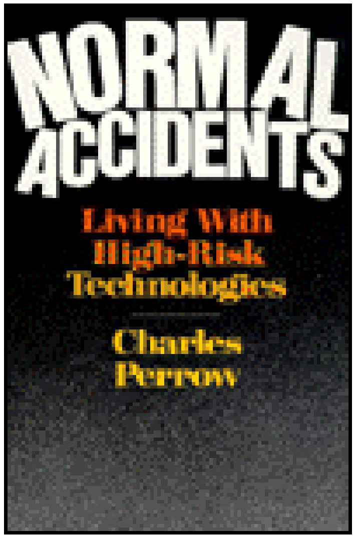
5 9s from Jim Gray's talk:
"Dependability
in the Internet Era"

"Microsoft fingers technicians for crippling site outages"

By Robert Lemos and Melanie Austria Farmer, ZDNet News, January 25, 2001

- Microsoft blamed its own technicians for a crucial error that crippled the software giant's connection to the Internet, almost completely blocking access to its major Web sites for nearly 24 hours… a "router configuration error" had caused requests for access to the company's Web sites to go unanswered…
- "This was an operational error and not the result of any issue with Microsoft or third-party products, nor with the securi works," a Microsoft spokesman said.
    - (5 9s possible if site stay rs!)

Learning from other fields: disasters



NORMAL ACCIDENTS

Living With High-Risk Technologies

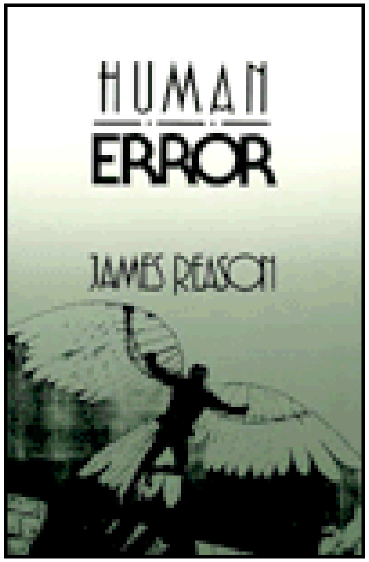
Charles Perrow

Common threads in accidents ~3 Mile Island

1. More multiple failures than you believe possible, because latent errors accumulate
2. Operators cannot fully understand system because errors in implementation, measurement system, warning systems. Also complex, hard to predict interactions
3. Tendency to blame operators afterwards (60-80%), but they must operate with missing, wrong information
4. The systems are never all working fully properly: bad warning lights, sensors out, things in repair
5. Emergency Systems are often flawed. At 3 Mile Island, 2 valves in wrong position; parts of a redundant system used only in an emergency. Facility running under normal operation masks errors in error handling

Source: Charles Perrow, *Normal Accidents: Living with High Risk Technologies*, Perseus Books, 1990

Learning from other fields: human error

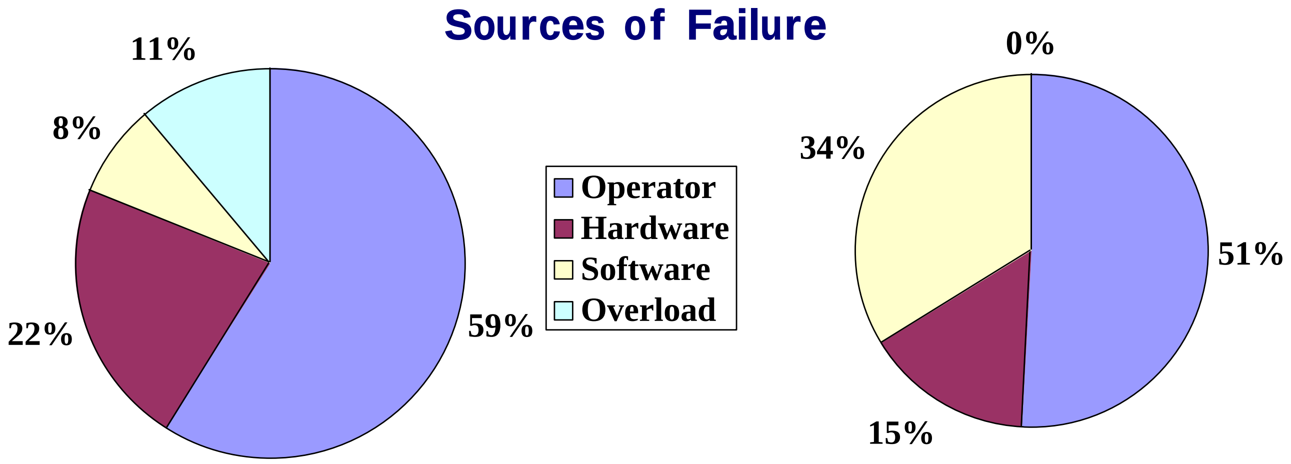


- **Two kinds of human error**
  - **1) slips/lapses:** errors in execution
  - **2) mistakes:** errors in planning
  - – errors can be **active** (operator error) or **latent** (design error, management error)
- **Human errors are inevitable**
  - – “humans are furious pattern-matchers”
    - » sometimes the match is wrong
  - – cognitive strain leads brain to think up least-effort solutions first, even if wrong
- **Humans can self-detect errors**
  - – about 75% of errors are immediately detected

Source: J. Reason, Human Error, Cambridge, 1990.

Human error

- **Human operator error is the leading cause of dependability problems in many domains**



Sources of Failure

Public Switched Telephone Network

Average of 3 Internet Sites

- **Operator error cannot be eliminated**
  - humans inevitably make mistakes: “to err is human”
  - *automation irony* tells us we can’t eliminate the human

Source: D. Patterson et al. Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies, UC Berkeley Technical Report UCB//CSD-02-1175, March 2002.

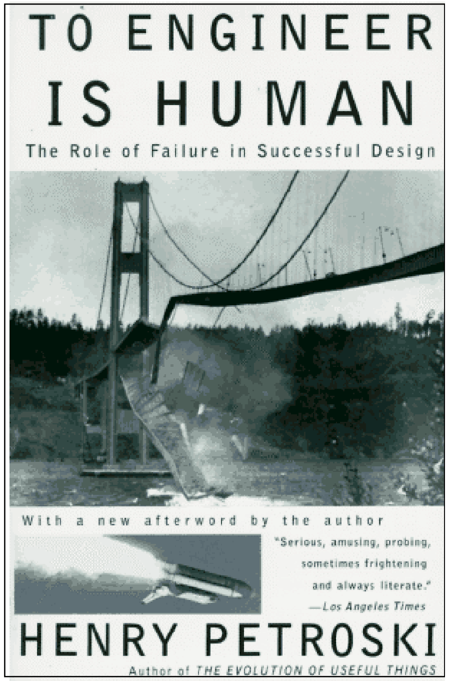
The ironies of automation

- **Automation doesn’t remove human influence**
  - shifts the burden from operator to designer
    - » designers are human too, and make mistakes
    - » unless designer is perfect, human operator still needed
- **Automation can make operator’s job harder**
  - reduces operator’s understanding of the system
    - » automation increases complexity, decreases visibility
    - » no opportunity to learn without day-to-day interaction
  - uninformed operator still has to solve exceptional scenarios missed by (imperfect) designers
    - » exceptional situations are already the most error-prone
- **Need tools to help, not replace, operator**

Source: J. Reason, Human Error, Cambridge University Press, 1990.

Learning from others: Bridges

- 1800s: 1/4 iron truss railroad bridges failed!
- Safety is now part of Civil Engineering DNA
- Techniques invented since 1800s:
  - Learn from failures vs. successes
  - Redundancy to survive some failures
  - Margin of safety 3X-6X vs. calculated load
  - (CS&E version of safety margin?)
- What will people of future think of our computers?


TO ENGINEER IS HUMAN
The Role of Failure in Successful Design
With a new afterword by the author
"Serious, amusing, probing, sometimes frightening and always literate."
—Los Angeles Times
HENRY PETROSKI
Author of THE EVOLUTION OF USEFUL THINGS

Margin of Safety in CS&E?

- **Like Civil Engineering, never make dependable systems until add margin of safety ("margin of ignorance") for what we don't (can't) know?**
  - Before: design to tolerate expected (HW) faults
- **RAID 5 Story**
  - Operator removing good disk vs. bad disk
  - Temperature, vibration causing failure before repair
  - In retrospect, suggested RAID 5 for what we anticipated, but should have suggested RAID 6 (double failure OK) for unanticipated/safety margin?
- **CS&S Margin of Safety: Tolerate human error in design, in construction, and in use?**

Where we are today

- **MAD TV, “Antiques Roadshow, 3005 AD”**

VALTREX:

“Ah ha. You paid 7 million Rubex too much. My suggestion: beam it directly into the disposal cube.

These pieces of crap crashed and froze so frequently that people became violent!

Hargh!”

“Worthless Piece of Crap: 0 Rubex”

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
- **The future:** how will history judge us?
- **Alternative future:** Recovery-Oriented Computing
- **ROC vs. Traditional Fault Tolerance**
- **ROC principles and quick examples**

A New Research Manifesto

- **Synergy with Humanity**
  - Build systems that work well with people who operate them, both end users on client computers and operators on server computers
- **Dependable Systems**
  - Build systems that world can safely depend upon
- **Secure Systems that Protect Privacy**
  - Need to help make society secure without compromising privacy of individuals
- **ROC project aimed at services at Internet sites, focus so far on synergy & dependability**

Recovery-Oriented Computing Philosophy

> "If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time"
>
> — *Shimon Peres ("Peres's Law")*

- **People/HW/SW failures are facts, not problems**
- **Recovery/repair is how we cope with them**
- **Improving recovery/repair improves availability**
  - UnAvailability = $\frac{\text{MTTR}}{\text{MTTF}}$ *(assuming MTTR much less than MTTF)*
  - 1/10th MTTR just as valuable as 10X MTBF
- **ROC also helps with maintenance/TCO**
  - since major Sys Admin job is recovery after failure
- **Since TCO is 5-10X HW/SW $, if necessary spend disk/DRAM/CPU resources for recovery**

MTTR more valuable than MTTF???

- **Threshold => non-linear return on improvement**
  - 8 to 11 second abandonment threshold on Internet
  - 30 second NFS client/server threshold
  - Satellite tracking and 10 minute vs. 2 minute MTTR
- **Ebay 4 hour outage, 1st major outage in year**
  - More people in single event worse for reputation?
  - One 4-hour outage/year => NY Times => stock?
  - What if 1-minute outage/day for a year?
    (250X improvement in MTTR, 365X worse in MTTF)
- **MTTF normally predicted vs. observed**
  - Include environmental error operator error, app bug?
  - Much easier to verify MTTR than MTTF!
- **If 99% to 99.9% availability, no change in prep**
  - 1-3 months => 10-30 months MTTF, still see failures

Traditional Fault-Tolerance vs.ROC

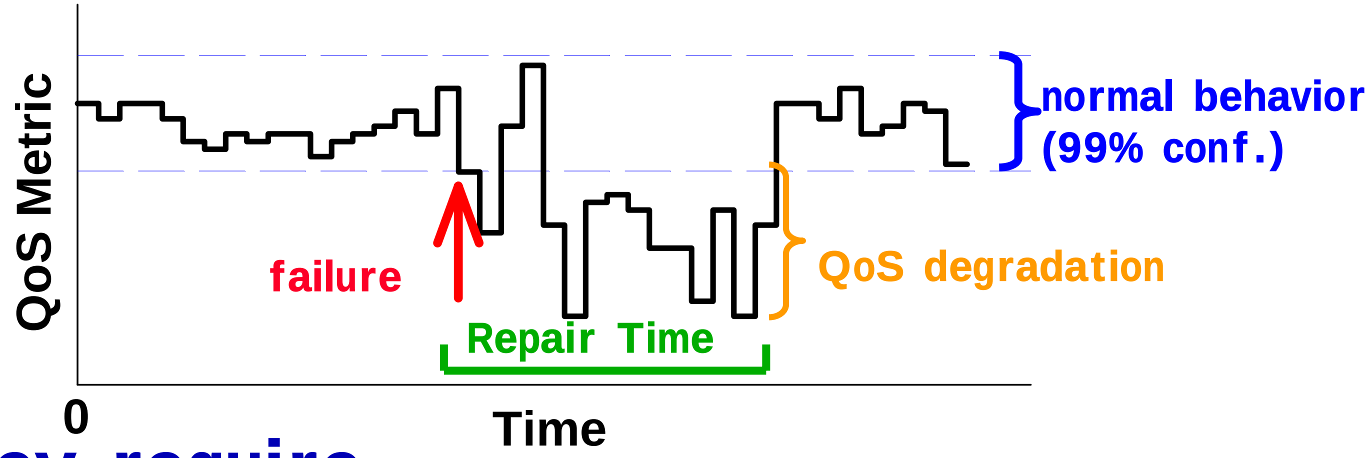
- **>30 years of Fault-Tolerance research**
  - fewer systems builders involved; ROC is for/by systems builders
- **FT greatest success in HW; ignores operator error?**
  - ROC holistic, all failure sources: HW, SW, and operator
- **FT tends to be bottom up, systems/ROC top-down**
- **Key FT approach: assumes accurate model of hardware and software, and ways HW and SW can fail**
  - Models to design, evaluate availability
  - Systems/ROC: benchmarks, quantitative evaluation of prototypes
- **Success areas for FT: airplanes, satellites, space shuttle, telecommunications, finance (Tandem)**
  - Hardware, software often changes slowly
  - Where SW/HW changes more rapidly, less impact of FT research
- **Much of FT helps MTTF, ROC helps MTTR**
  - Improving MTTF and MTTR synergistic (don’t want bad MTTF!)

Five "ROC Solid" Principles

1. **Given errors occur, design to recover rapidly**
2. **Given humans make errors, build tools to help operator find and repair problems**
   - e.g., undo; hot swap; graceful, gradual SW upgrade
3. **Extensive sanity checks during operation**
   - To discover failures quickly (and to help debug)
   - Report to operator (and remotely to developers)
4. **Any error message in HW or SW can be routinely invoked, scripted for regression test**
   - To test emergency routines during development
   - To validate emergency routines in field
   - To train operators in field
5. **Recovery benchmarks to measure progress**
   - Recreate performance benchmark competition

Recovery benchmarking 101

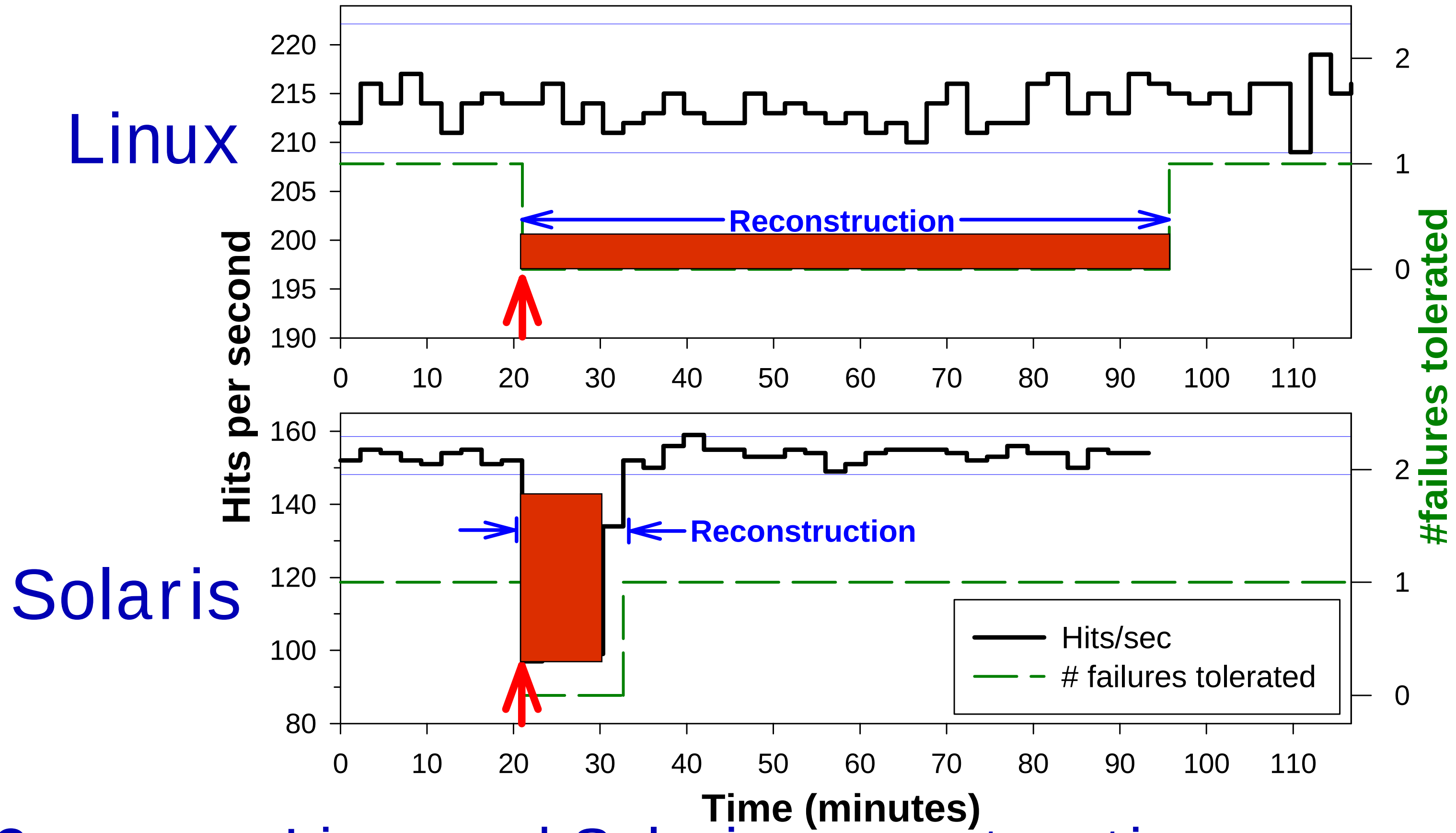
- **Recovery benchmarks quantify system behavior under failures, maintenance, recovery**



- **They require**
  - A realistic workload for the system
  - Quality of service metrics and tools to measure them
  - Fault-injection to simulate failures
  - Human operators to perform repairs

Source: A. Brown, and D. Patterson, “Towards availability benchmarks: a case study of software RAID systems,” *Proc. USENIX*, 18-23 June 2000

Example: 1 fault in SW RAID



- Compares Linux and Solaris reconstruction
  - **Linux:** Small impact but longer vulnerability to 2nd fault
  - **Solaris:** large perf. impact but restores redundancy fast
  - **Windows**: did not auto-reconstruct!

Recovery Benchmarks (so far)

- **Recovery benchmarks involve people, but so do most research by social scientists**
  - "Macro" benchmarks for competition, must be fair, hard to game, representative; use ~ 10 operators in routine maintenance and observe errors; insert realistic HW, SW errors stochastically
  - "Micro" benchmarks for development, must be cheap; inject typical human, HW, SW errors; predict Macro
- **Many opportunities to compare commercial products and claims, measure value of research ideas, … with recovery benchmarks**
  - Lots of low hanging fruit (~ early RAID days)

Source: D. Oppenheimer, A. Brown, J. Traupman, P. Broadwell, and D. Patterson. Practical issues in dependability benchmarking. 2nd Workshop on Evaluating and Architecting System Dependability (EASY), Oct. 2002

Help Operator with Diagnosis?

- **System assists human in diagnosing problems**
  - Root-cause analysis to suggest possible failure points
    - » Track resource dependencies of all requests
    - » Correlate symptomatic requests with component dependency model to isolate culprit components
  - "health" reporting to detect failed/failing components
    - » Failure information, self-test results propagated upwards
  - Don't rely on things connected according to plans
    - » Example: Discovery of network, power topology
- **Example: Pinpoint – modify J2EE to trace modules used and record success/fail of trace, then use standard data mining to discover failed module; 8% overhead, don't need model, yet very accurate**

Source: Chen, M., E. Kiciman, E. Fratkin, E. Brewer and A. Fox. Pinpoint: Problem Determination in Large, Dynamic, Internet Services. Proc. Int'l Conf. on Dependable Systems and Networks, Washington D.C., 2002.

Support Operator Repair?

- **Time travel for system operators for high level commands**
- **Three R's for recovery**
  - **Rewind:** roll all system state backwards in time
  - **Repair:** change system to prevent failure
    - » e.g., fix latent error, retry unsuccessful operation, install preventative patch
  - **Replay:** roll system state forward, replaying end-user interactions lost during rewind
- **All three R's are critical**
  - rewind enables undo
  - repair lets user/administrator fix problems
  - replay preserves updates, propagates fixes forward

Example 3R’s scenarios

- **Retroactive repair**
  - – mitigate external attacks
    - » retroactively install virus/spam filter on email server; effects are squashed on replay
- **Undo spends excess disk capacity to offer safety margin via time travel => versioning file system, log of email events, ..**
- **(Recent) Key Insight: leverage file consistency research for disconnected users (e.g.,Bayou)**
  - – file systems modified in parallel, later “synced”

Source: A. Brown, and D. A. Patterson. Rewind, Repair, Replay: Three R's to Dependability. 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002.

Error Insertion Example?

- **Example: FIG - Fault Insertion in Glibc**
  - <10% overhead in portable library
  - finds strange behavior even in mature software when invoke errors
  - Code is available

Source: Broadwell, P., N. Sastry and J. Traupman. FIG: A Prototype Tool for Online Verification of Recovery Mechanisms. Workshop on Self-Healing, Adaptive and self-MANaged Systems (SHAMAN), New York, NY, June 2002.

Rapid Recovery via Recursive Restart?

- **“Recursive Recovery” (Candea, Fox) restarts optimal number of components of system**
- **Look at dependence chain during recovery to see if can reorganize to reduce recovery time**
- **Example: Mercury satellite ground station**
  - Average 5X reduction in recovery time
  - Nonlinear return: fast recovery implies don’t lose track of satellite during pass vs. greater MTTF

Source: G. Candea and A. Fox, “Recursive Restartability: Turing the Reboot Sledgehammer into a scalpel,” ***8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*****, May 2001**

ROC Status

- Papers that layout philosophy and initial results for
  - Recovery benchmarks
  - Failure data collection and analysis
  - Error insertion
  - Diagnosis without detailed model
  - MTTR v. MTTF
  - Fast recovery
  - Undo design and implementation
- Building Email prototype for operator undo
- Plan on Email system using all ROC techniques, then benchmark recovery vs. commercial systems

ROC Summary, Part I

- **Need a theory on constructing dependable, maintainable sites for networked services**
  - Document best practices of successful sites?
- **Need a theory on good design for <u>operators</u> as well as good design for <u>end users</u>**
  - Airplane Analogy: user interface to passengers (747) vs. user interface to pilots (Cessna)
  - HCI research opportunity?
- **Need new definition of "performability"**
  - Failure is more than unavailable for 100% of users: (e.g., available to 10% of users is not "up")
  - Cost of outages to Internet service like cost of overloads: customers give up, income lost
  - Need IT equivalent of PSTN "blocked calls"?
    - » PSTN switches required to collect blocked calls

Cautionary Tale

- **Motivation #1**: We should build dependable, secure systems that are synergistic with humanity because computer scientists and engineers are moral people and we know it’s the right thing to do
- **Motivation #2**: Governments will soon enable litigation against undependable, insecure products that crash and freeze so frequently that people become violent

ROC Summary, Part II

- 21st Century Research challenge is Synergy with Humanity, Dependability, Security/Privacy
- CS&E Margin of Safety: Tolerate Human Error?
- 2002: Peres's Law greater than Moore's Law?
  - Must cope with fact that people, SW, HW fail
- Recovery Oriented Computing is one path for operator synergy, dependability for servers
  - Failure data collection + Benchmarks to evaluate
  - Industry: may soon compete on recovery time v. SPEC
  - Undo support, Error Insertion, Sanity Checks, Recursive Recovery, Diagnosis Aid,
  - Significantly reducing MTTR (people/SW/HW) => better Dependability & lower Cost of Ownership

Interested in ROCing?

- **More research opportunities than 2 university projects can cover. Many could help with:**
  - Failure data collection, analysis, and publication
  - Create/Run Recovery benchmarks: compare (by vendor) databases, files systems, routers, …
  - Invent, evaluate techniques to reduce MTTR and TCO in computation, storage, and network systems
  - (Lots of low hanging fruit)

“If it’s important,
how can you say it’s impossible if you don’t try?”

Jean Monnet, a founder of European Union

http://ROC.cs.berkeley.edu

BACKUP SLIDES

Recovery Benchmarking Environment

- **Fault workload**
  - – Must accurately reflect failure modes of real-world Internet service environments
    - » plus random tests to increase coverage, simulate Heisenbugs
  - – But, no existing public failure dataset
    - » we have to collect this data
    - » a challenge due to proprietary nature of data
  - – major contribution will be to collect, anonymize, and publish a modern set of failure data
- **Fault injection harness**
  - – build into system: needed anyway for online verification

Safe, forgiving for operator?

- **Expect human error and tolerate it**
  - protect system data from human error
  - allow mistakes to be easily reversed
- **Allow human operator to learn naturally**
  - “mistakes are OK”: design to encourage exploration, experimentation
- **Make training on real system an everyday process**
- **Match interfaces to human capabilities**
- **Automate tedious or difficult tasks, but retain manual procedures**
  - Encourage periodic use of manual procedures to increase familiarity

Automation vs. Aid?

- **Two approaches to helping**

1) Automate the entire process as a unit

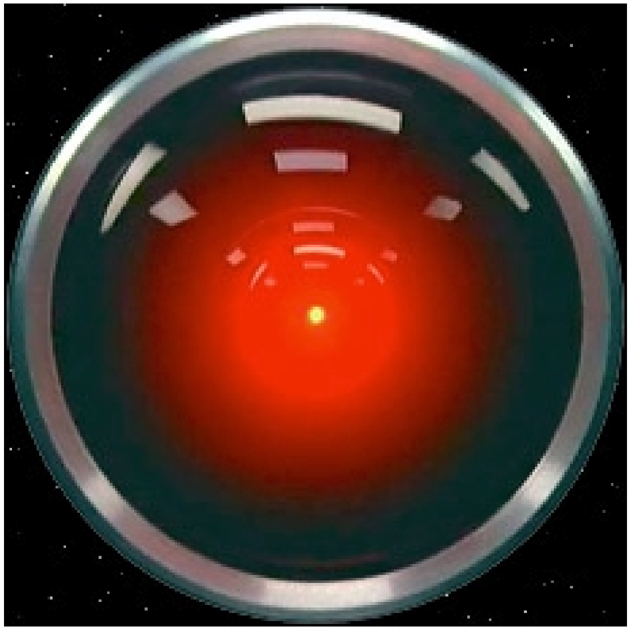
- – the goal of most research into “self-healing”, “self-maintaining”, “self-tuning”, or more recently “introspective” or “autonomic” systems
- – What about Automation Irony?

2) ROC approach: provide tools to let human SysAdmins perform job more effectively

- – If desired, add automation as a layer on top of the tools
- – What about number of SysAdmins as number of computers continue to increase?

A science fiction analogy

- **Full automation**



HAL 9000 (2001)

- **Suffers from effects of the automation ironies**
  - system is opaque to humans
  - only solution to unanticipated failure is to pull the plug?

- **Human-aware automation**

Enterprise computer (2365)

- **24th-century engineer is like today's SysAdmin**
  - a *human* diagnoses & repairs computer problems
  - automation used in human-operated diagnostic tools

Challenge #2: externalized state

- **The equivalent of the “time travel paradox”**
  - the 3R cycle alters state that has previously been seen by an external entity (user or another computer)
  - produces inconsistencies between internal and external views of state after 3R cycle
- **Examples**
  - a formerly-read/forwarded email message is altered
  - a failed request is now successful or vice versa
  - item availability estimates change in e-commerce, affecting orders
- **No complete fix; solutions just manage the inconsistency**

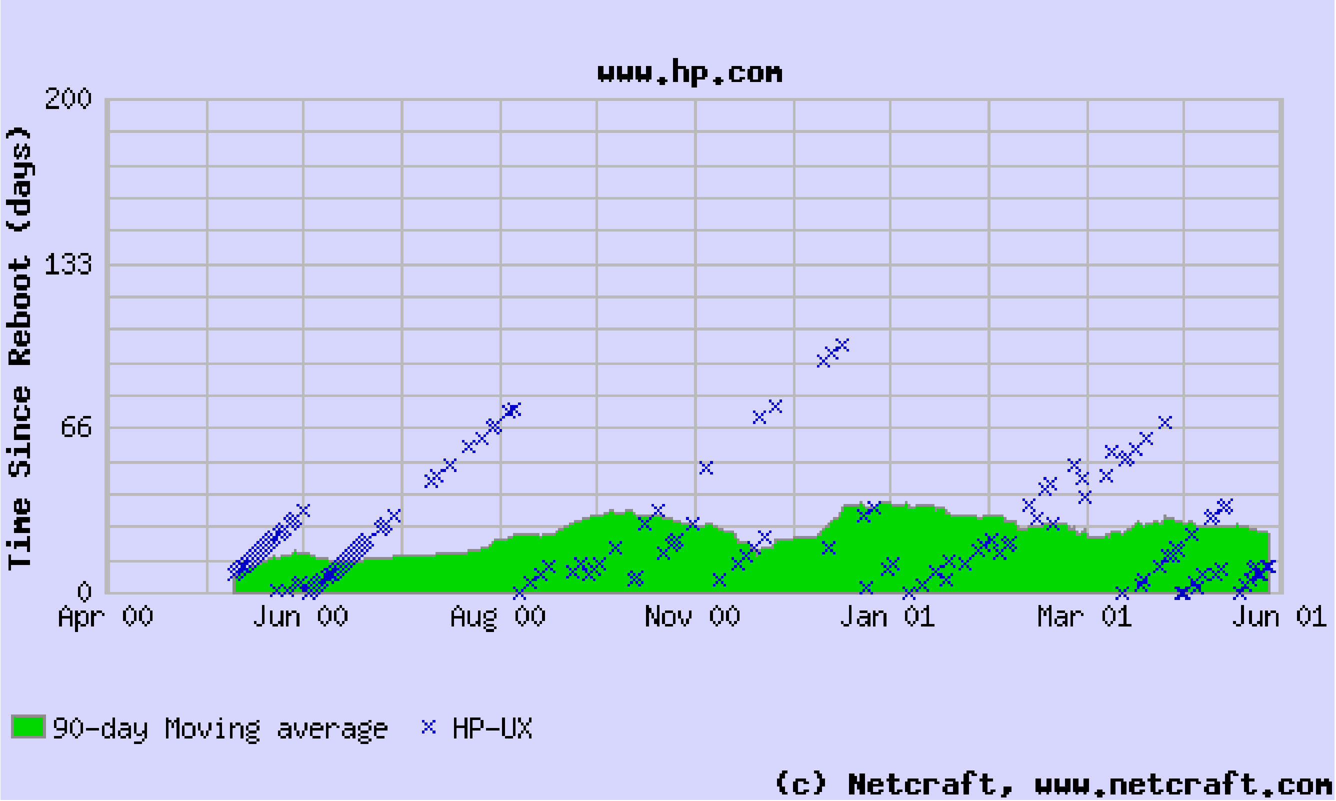
Externalized state: solutions

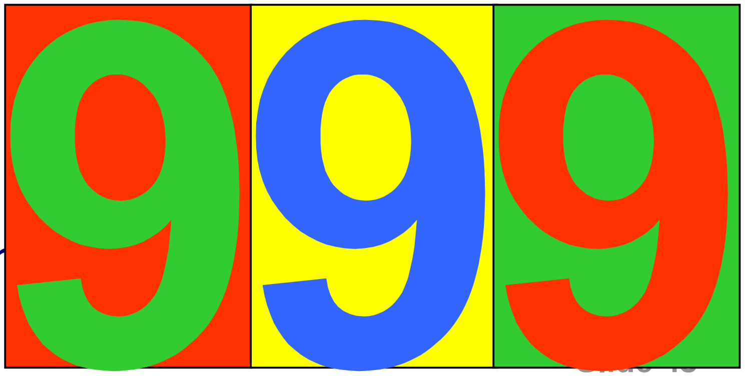
- **Ignore the inconsistency**
  - – let the (human) user tolerate it
  - – appropriate where app. already has loose consistency
    - » *e.g.*, email message ordering, e-commerce stock estimates
- **Compensating/explanatory actions**
  - – leave the inconsistency, but explain it to the user
  - – appropriate where inconsistency causes confusion but not damage
    - » *e.g.*, 3R's delete an externalized email message; compensating action replaces message with a new message explaining why the original is gone
    - » *e.g.*, 3R's cause an e-commerce order to be cancelled; compensating action refunds credit card and emails user

Externalized state: solutions (2)

- **Expand the boundary of Rewind**
  - 3R cycle induces rollback of external system as well
    - external system reprocesses updated externalized data
  - appropriate when externalized state chain is short; external system is under same administrative domain
    - danger of expensive cascading rollbacks; exploitation
- **Delay execution of externalizing actions**
  - allow inconsistency-free undo only within delay window
  - appropriate for asynchronous, non-time-critical events
    - *e.g.*, sending mailer-daemon responses in email or delivering email to external hosts

Availability: Uptime of HP.com?





- **Average reboot is about 30.8**
  if 10 minutes per reboot => 9
  - See uptime.netcraft.com/up/graph

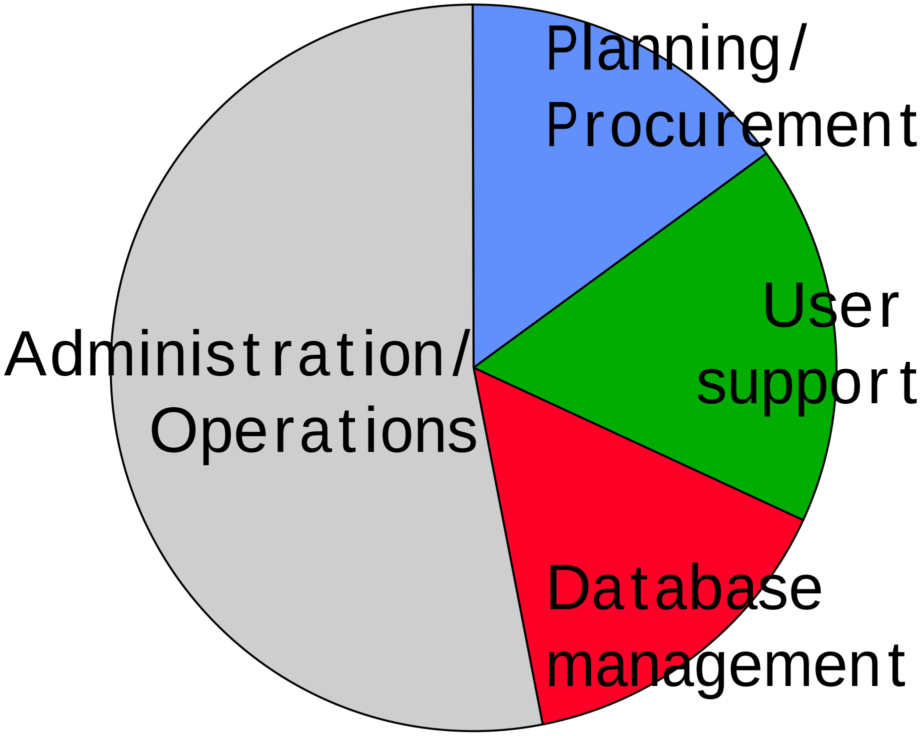
Software RAID: QoS behavior

- **Response to double-fault scenario**
  - a double fault results in unrecoverable loss of data on the RAID volume
  - **Linux:** blocked access to volume
  - **Windows:** blocked access to volume
  - **Solaris:** silently continued using volume, delivering *fabricated* data to application!
    - » clear violation of RAID availability semantics
    - » resulted in corrupted file system and garbage data at the application level
    - » this *undocumented* policy has serious availability implications for applications

Partitioning and Redundancy?

- **System is Partitionable**
  - To isolate faults
  - To enable online repair/recovery
  - To enable online HW growth/SW upgrade
  - To enable operator training/expand experience on portions of real system without fear of system failure
  - Techniques: Geographically replicated sites, Virtual Machine Monitors
- **System is Redundant**
  - Sufficient HW redundancy/Data replication => part of system down but satisfactory service still available
  - Enough to survive 2nd (nth?) failure during recovery
  - Techniques: RAID-6, N-copies of data

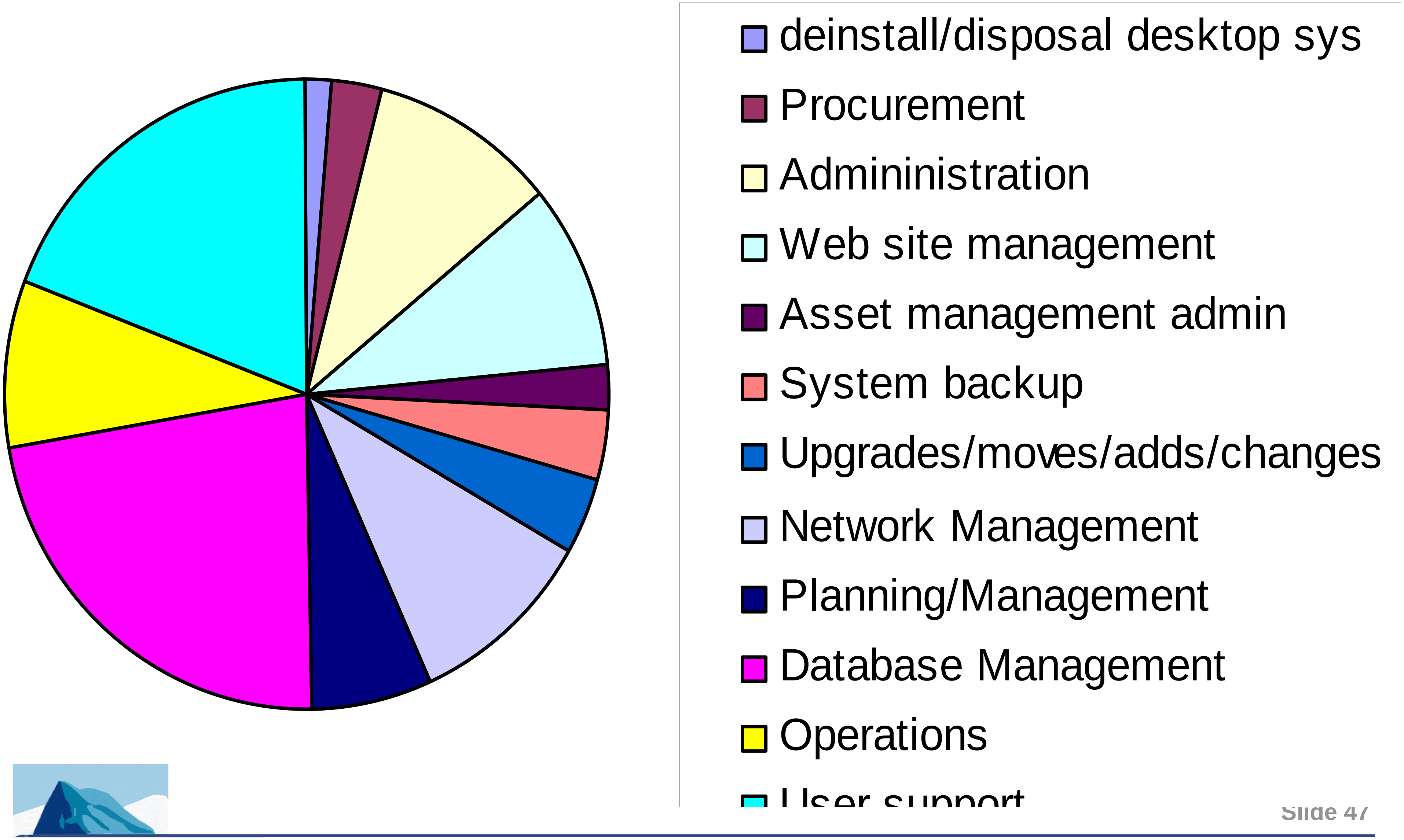
TCO breakdown (average)



- **Administration/Operations**
  - Adding/deleing users
  - Tracking equipment
  - Network, Server management
  - Backup
  - Upgrades, Web site
- **Planning/Procurement**
  - Planning for upgrades
  - Buying new, disposing old
- **User support**
  - Help desk
  - Desktop troubleshooting
- **Database management**
  - Creating, adjusting, allocating DB resources

Source: "The Role of Linux in Reducing the Cost of Enterprise Computing", IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com

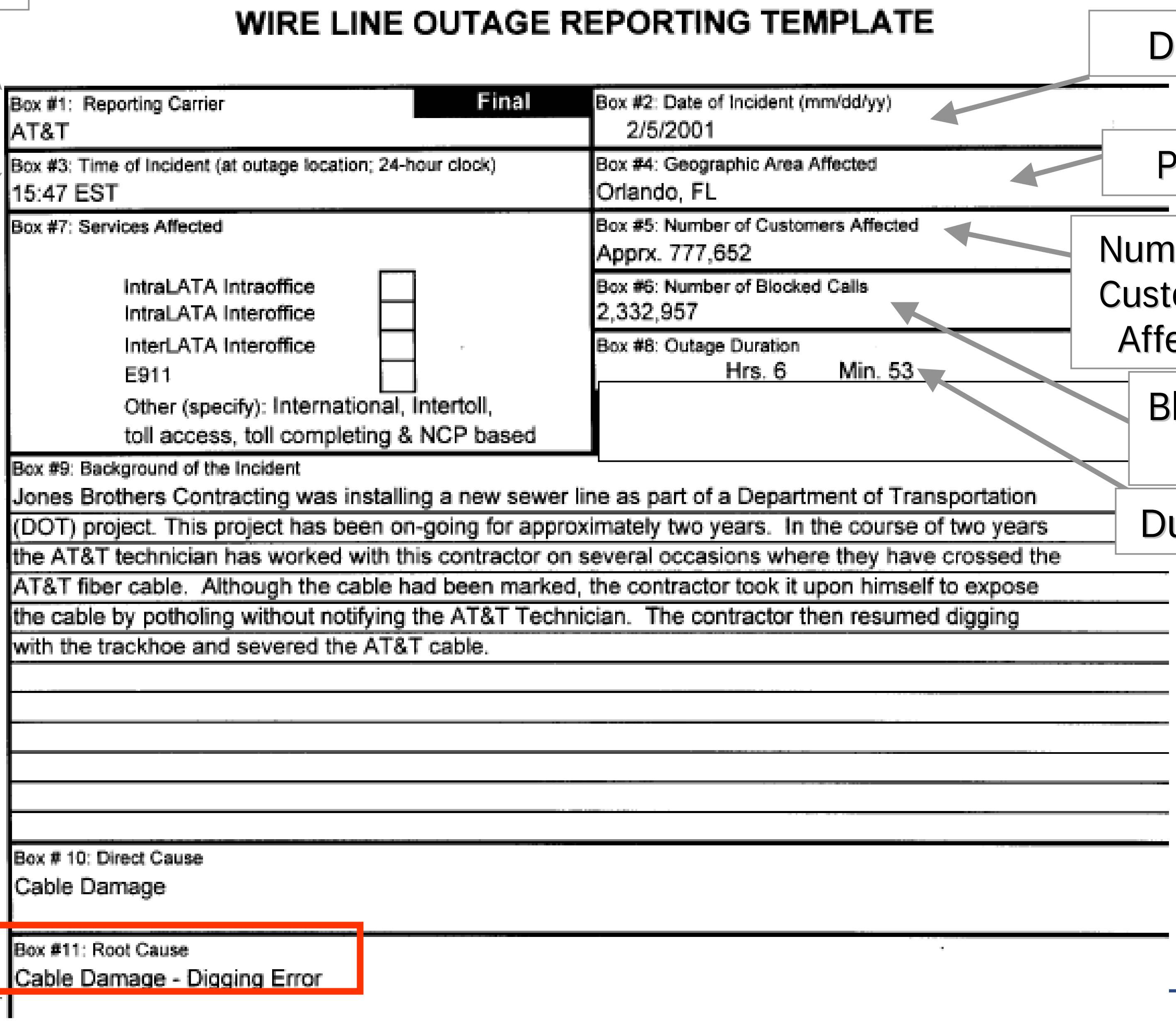
Internet x86/Linux Breakdown



Total Cost Own. Hypothesis

- **“Moore’s Law” + hypercompetitve marketplace improves cost and speed of CPUs, cost and capacity of memory and disks**
- **Morris (IBM) $3M comparison 1984 v. 2001:**
  - CPU: Minicomputer to PC, 3000X faster
  - DRAM: Memory boards to DIMMs, 3000X bigger
  - Disks: 8-inch drives to 3.5-inch drives, 4000X bigger
- **Unless avg. user demands grow with Moore’s Law, a service increases in number of users**
- **HW/SW costs shrink; salaries go up over time**
- **Hypothesis: Cost of Ownership is more a function of number of users versus HW/SW $, so T.C.O. today is mostly people costs**

Outage Report



WIRE LINE OUTAGE REPORTING TEMPLATE

Box #1: Reporting Carrier **Final** AT&T	Box #2: Date of Incident (mm/dd/yy) 2/5/2001
Box #3: Time of Incident (at outage location; 24-hour clock) 15:47 EST	Box #4: Geographic Area Affected Orlando, FL
Box #7: Services Affected IntraLATA Intraoffice □ IntraLATA Interoffice □ InterLATA Interoffice □ E911 □ Other (specify): International, Intertoll, toll access, toll completing & NCP based	Box #5: Number of Customers Affected Apprx. 777,652 Box #6: Number of Blocked Calls 2,332,957 Box #8: Outage Duration Hrs. 6 Min. 53

Box #9: Background of the Incident

Jones Brothers Contracting was installing a new sewer line as part of a Department of Transportation (DOT) project. This project has been on-going for approximately two years. In the course of two years the AT&T technician has worked with this contractor on several occasions where they have crossed the AT&T fiber cable. Although the cable had been marked, the contractor took it upon himself to expose the cable by potholing without notifying the AT&T Technician. The contractor then resumed digging with the trackhoe and severed the AT&T cable.

Box # 10: Direct Cause

Cable Damage

Box #11: Root Cause

Cable Damage - Digging Error

Company

Date

Place

Time

Number of Customers Affected

Blocked Calls

Duration

Explanation

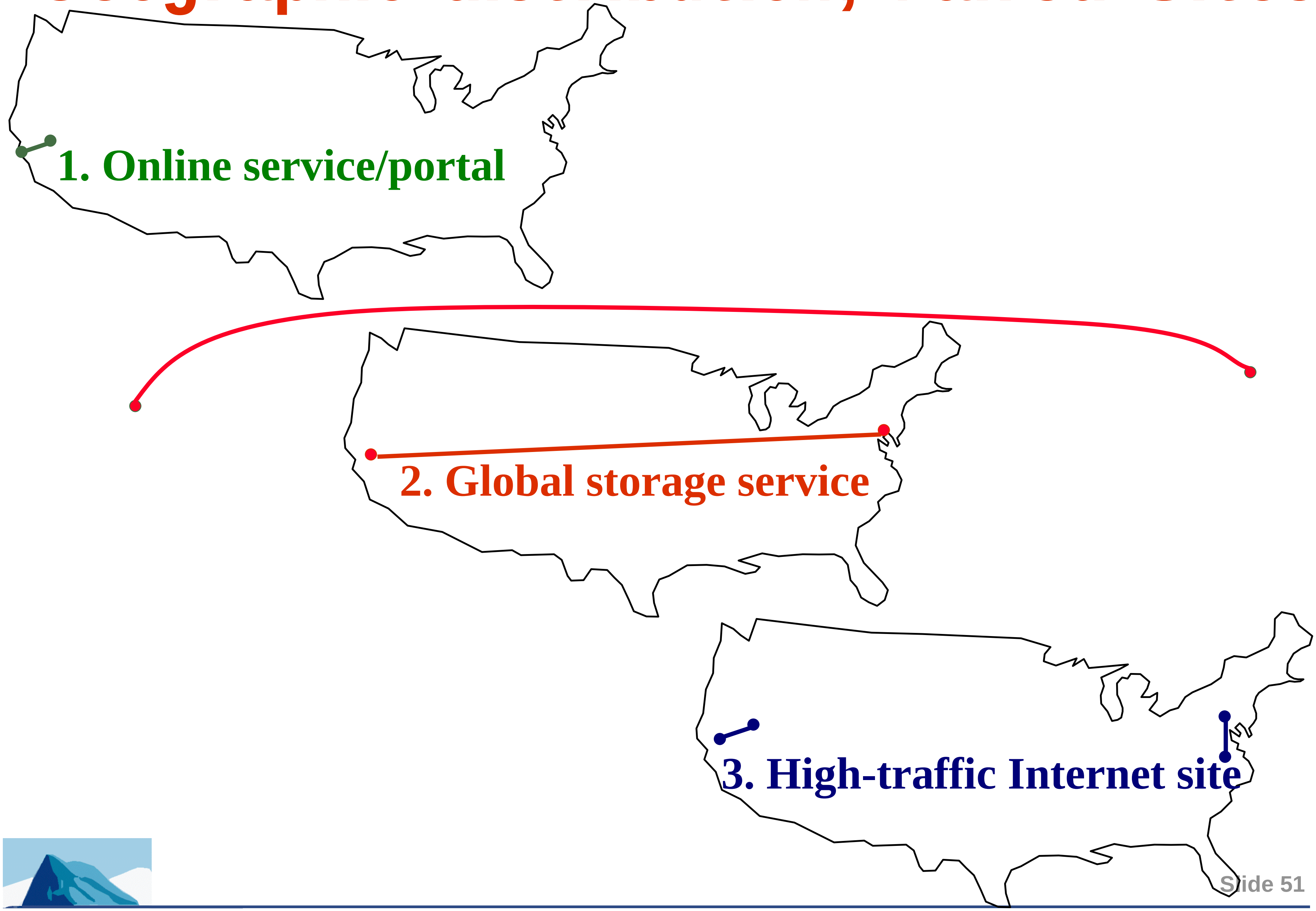
Cause

Failure Data: 3 Internet Sites

- **Global storage service site**
  - ~500 machines, 4 colo. facilities + customer sites
  - all service software custom-written (x86/free OS)
- **High-traffic Internet site**
  - ~5000 of machines, 4 collocation facilities
  - ~100 million hits/day
  - all service software custom-written (x86/free OS)
  - Read mostly
- **Online services site**
  - R/W, ~1000 machines, custom SW, Sparc/x86 Solaris
- **Looked at trouble tickets over 3-6 months**

Source: David Oppenheimer, U.C. Berkeley, in progress.

Geographic distribution, Paired Sites



1. Online service/portal

2. Global storage service

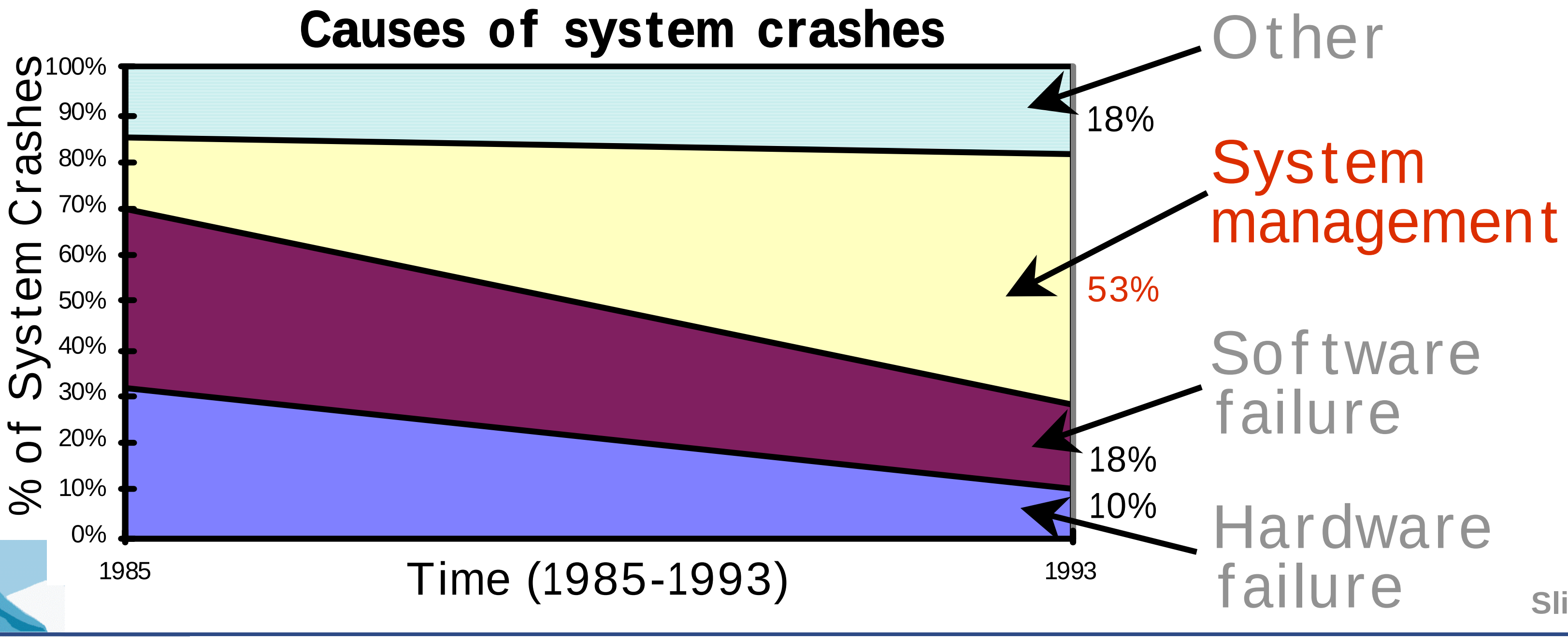
3. High-traffic Internet site

Evaluating ROC: human aspects

- **Must include humans in availability benchmarks**
  - to verify effectiveness of undo, training, diagnostics
  - humans act as system administrators
- **Subjects should be admin-savvy**
  - system administrators
  - CS graduate students
- **Challenge will be compressing timescale**
  - i.e., for evaluating training
- **We have some experience with these trials**
  - earlier work in maintainability benchmarks used 5-person pilot study

ROC Part I: Failure Data
Lessons about human operators

- **Human error is largest single failure source**
  - HP HA labs: human error is #1 cause of failures (2001)
  - Oracle: half of DB failures due to human error (1999)
  - Gray/Tandem: 42% of failures from human administrator errors (1986)
  - Murphy/Gent study of VAX systems (1993):



Lessons Learned from Other Cultures

- Code of Hammurabi, 1795-1750 BC, Babylon
  - 282 Laws on 8-foot stone monolith

229. If a builder build a house for some one, and does not construct it properly, and the house which he built fall in and kill its owner, then that builder shall be put to death.

230. If it kill the son of the owner the son of that builder shall be put to death.

232. If it ruin goods, he shall make compensation for all that has been ruined, and inasmuch as he did not construct properly this house which he built and it fell, he shall re-erect the house from his own means.

- Do we need Babylonian quality standards?

Butler Lampson: Systems Challenges

- **Systems that work**
  - Meeting their specs
  - Always available
  - Adapting to changing environment
  - Evolving while they run
  - Made from unreliable components
  - Growing without practical limit
- **Credible simulations or analysis**
- **Writing good specs**
- **Testing**
- **Performance**
  - Understanding when it doesn’t matter

“Computer Systems Research
-Past and Future”
Keynote address,
17th SOSP,
Dec. 1999
Butler Lampson
Microsoft