Recovery-Oriented Computing

Dave Patterson and Aaron Brown

University of California at Berkeley

{patterson,abrown}@cs.berkeley.edu

In cooperation with

Armando Fox, Stanford University

fox@cs.stanford.edu

http://roc.CS.Berkeley.EDU/

RECOVERY-ORIENTED COMPUTING

October 2001

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
- **The future:** Recovery-Oriented Computing (ROC)
- **ROC techniques and principles**

RECOVERY-ORIENTED COMPUTING

The past: goals and assumptions of last 15 years

- **Goal #1: Improve performance**
- **Goal #2: Improve performance**
- **Goal #3: Improve cost-performance**
- **Assumptions**
  - Humans are perfect (they don't make mistakes during installation, wiring, upgrade, maintenance or repair)
  - Software will eventually be bug free (good programmers write bug-free code, debugging works)
  - Hardware MTBF is already very large (~100 years between failures), and will continue to increase

Recovery-Oriented Computing

Today, after 15 years of improving performance

- **Availability is now the vital metric for servers**
  - near-100% availability is becoming mandatory
    - » for e-commerce, enterprise apps, online services, ISPs
  - but, service outages are frequent
    - » 65% of IT managers report that their websites were unavailable to customers over a 6-month period
      - 25%: 3 or more outages
  - outage costs are high
    - » social effects: negative press, loss of customers who “click over” to competitor

Source: InternetWeek 4/3/2000

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."

What have we learned from past projects?

- **Maintenance of machines (with state) expensive**
  - ~5X to 10X cost of HW
  - Stateless machines can be trivial to maintain (Hotmail)
- **System admin primarily keeps system available**
  - System + clever human working during failure = uptime
  - Also plan for growth, software upgrades, configuration, fix performance bugs, do backup
- **Know how evaluate (performance and cost)**
  - Run system against workload, measure, innovate, repeat
  - Benchmarks standardize workloads, lead to competition, evaluate alternatives; turns debates into numbers
    - **What are the new challenges? Says who?**

Recovery-Oriented Computing

Jim Gray: Trouble-Free Systems

"What Next?
A dozen remaining IT problems"
Turing Award Lecture,
FCRC,
May 1999
Jim Gray
Microsoft

- **Manager**
  - Sets goals
  - Sets policy
  - Sets budget
  - System does the rest.
- **Everyone is a CIO (Chief Information Officer)**
- **Build a system**
  - Used by millions of people each day
  - Administered and managed by a ½ time person.
    - » On hardware fault, order replacement part
    - » On overload, order additional equipment
    - » Upgrade hardware and software automatically.

Recovery-Oriented Computing

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

Recovery-Oriented Computing

John Hennessy: What Should the "New World" Focus Be?

- Availability
  - Both appliance & service
- Maintainability
  - Two functions:
    - » Enhancing availability by preventing failure
    - » Ease of SW and HW upgrades
- Scalability
  - Especially of service
- Cost
  - per device and per service transaction
- Performance
  - Remains important, but its not SPECint

"Back to the Future: Time to Return to Longstanding Problems in Computer Systems?"
Keynote address, FCRC, May 1999
John Hennessy Stanford

RECOVERY-ORIENTED COMPUTING

Charlie Bell, Amazon.com (Monday)

- **Goals of Internet commerce system design:**
  - Support Change: rapid innovation
    - » "each service can be updated every few days"
  - Unconstrained scalability
  - Always-on availability
  - Latency for outliers is the performance metric

Common goals: ACME

- **Availability**
  - 24x7 delivery of service to users
- **Change**
  - support rapid deployment of new software, apps, UI
- **Maintainability**
  - reduce burden on system administrators
  - provide helpful, forgiving sysadmin environments
- **Evolutionary Growth**
  - allow easy system expansion over time without sacrificing availability or maintainability

Where does ACME stand today?

- **Availability: failures are common**
  - Traditional fault-tolerance doesn't solve the problems
- **Change**
  - In back-end system tiers, software upgrades difficult, failure-prone, or ignored
  - For application service over WWW, daily change
- **Maintainability**
  - human operator error is single largest failure source
  - system maintenance environments are unforgiving
- **Evolutionary growth**
  - 1U-PC cluster front-ends scale, evolve well
  - back-end scalability still limited

Recovery-Oriented Computing

ACME: Availability

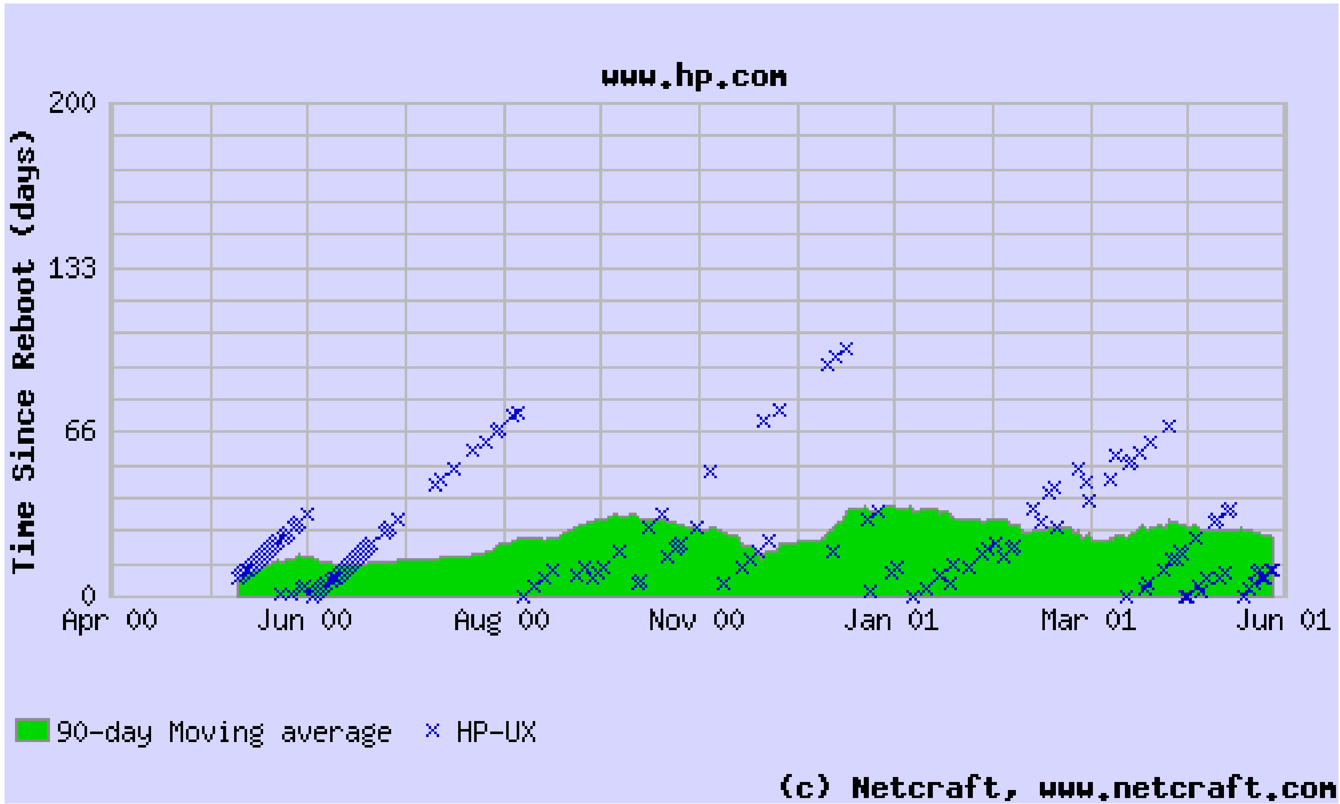
- **Availability: failures are common**
  - Well designed and manufactured HW: >1% fail/year
  - Well designed and tested SW: > 1 bug / 1000 lines
  - Well trained people doing difficult tasks: up to 10%
  - Well run co-location site (e.g., Exodus): 1 power failure per year, 1 network outage per year
  - Denial of service attacks => routine event

ACME: What about claims of 5 9s?

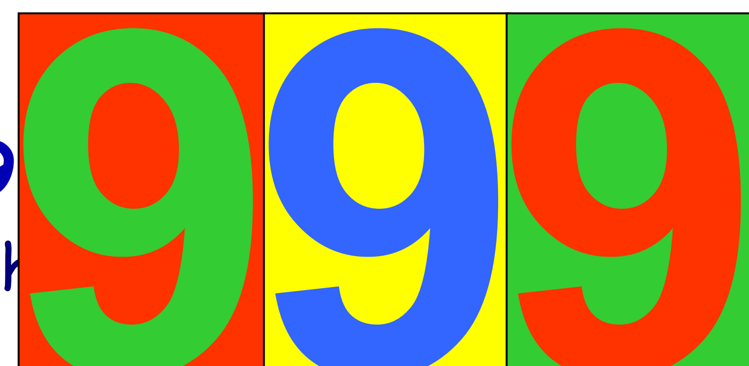
- 99.999% availability from telephone company?
  - **AT&T switches < 2 hours of failure in 40 years**
- Cisco, HP, Microsoft, Sun … claim 99.999% availability claims (5 minutes down / year) in marketing/advertising
  - **HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee "in certain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

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

ACME: What is uptime of HP.com?



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

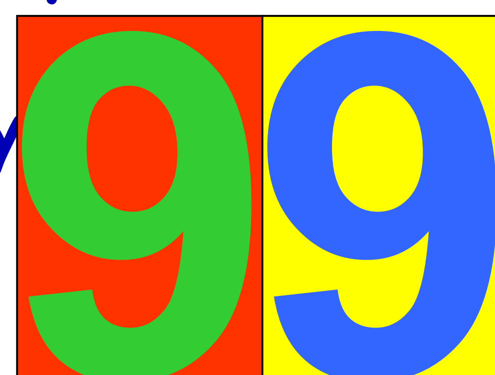


Recovery-Oriented Computing

"Microsoft fingers technicians for crippling site outages"

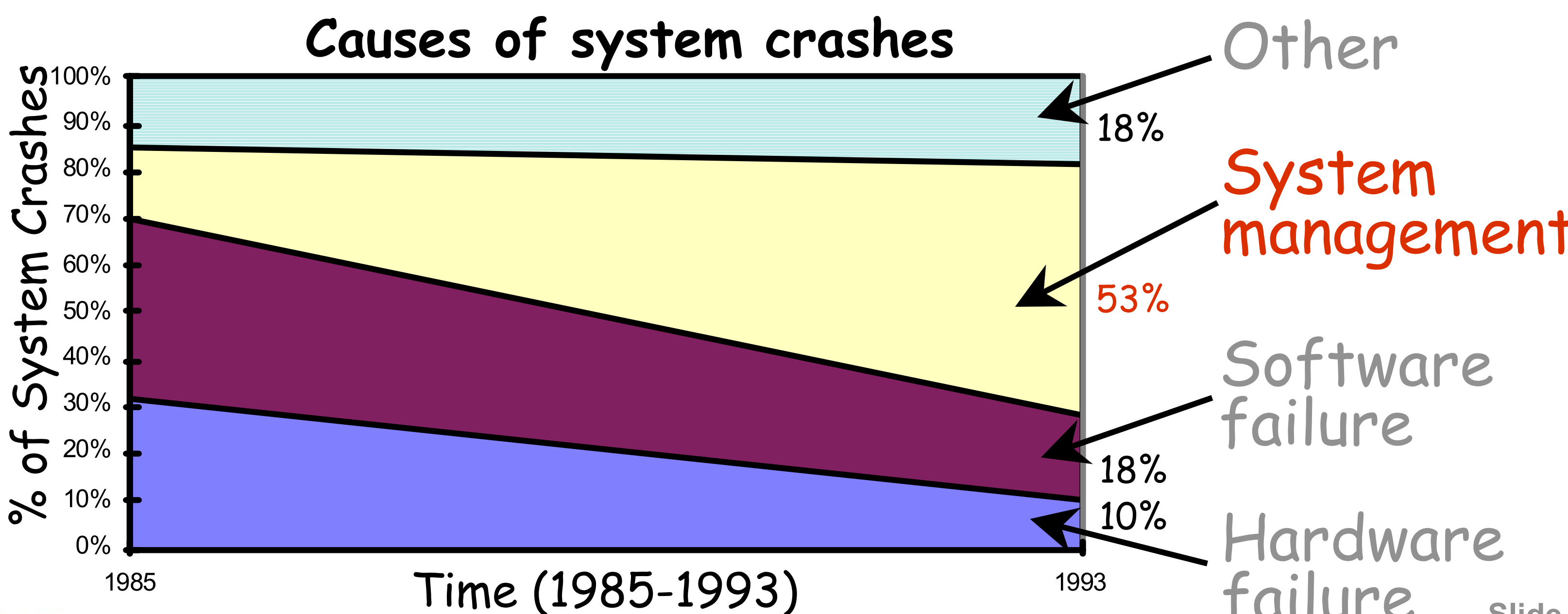
By Robert Lemos and Melanie Austria Farmer, ZDNet News, January 25, 2001

- 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 security of our networks," a Microsoft spokesman said.
  - (5 9s possible if site stay rs!)



ACME: 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):

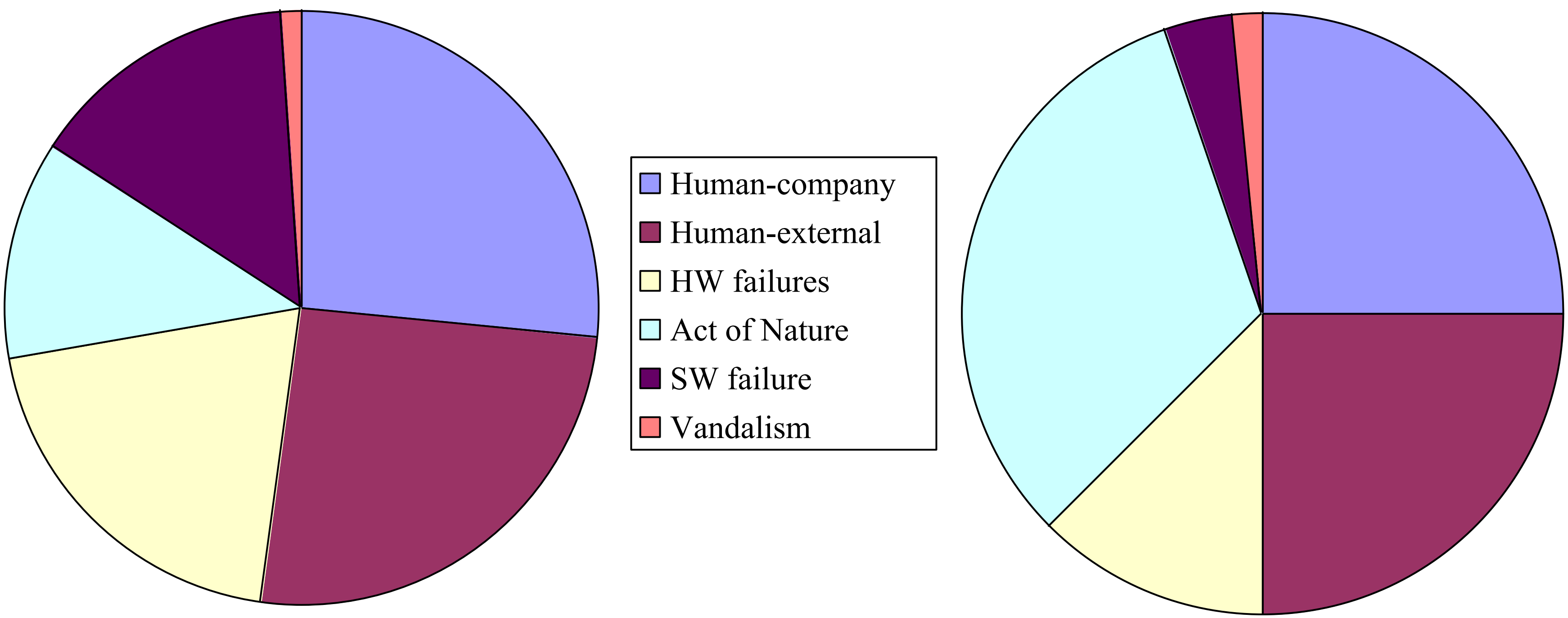


Recovery-Oriented Computing

ACME: Learning from other fields: PSTN

- **Causes of telephone network outages**
  - from FCC records, 1992-1994

Number of Outages | **Number customers x Minutes of Failure**



  - half of outages, outage-minutes are human-related
    - » about 25% are direct result of maintenance errors by phone company workers

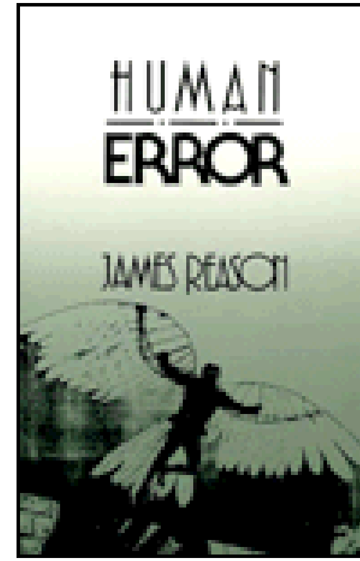
Source: Kuhn, IEEE Computer 30(4), 1997.

ACME: Trends in Customer Minutes 1992-94 vs. 2001

Cause	Trend	Minutes (millions of customer minutes/month) 1992-94	2001
Human Error: Company	↑	98	176
Human Error: External	↓	100	75
Hardware	↔	49	49
Software	↓	15	12
Overload	↓	314	60
Vandalism	↓	5	3

Recovery-Oriented Computing

ACME: Learning from other fields: human error



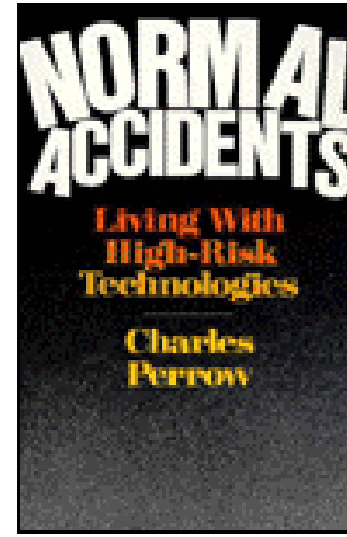
- **Two kinds of human error**
  - **1) slips/lapses:** errors in execution
  - **2) mistakes:** errors in planning
  - errors can be **active** (operator error) or **latent** (design error, management error)
- **Human errors are inevitable**
  - "humans are furious pattern-matchers"
    - » sometimes 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.

ACME: The Automation Irony

- **Automation does not cure human error**
  - automation addresses the easy tasks, leaving the complex, unfamiliar tasks for the human
    - » humans are ill-suited to these tasks, especially under stress
  - automation hinders understanding and mental modeling
    - » decreases system visibility and increases complexity
    - » operators don't get hands-on control experience
    - » prevents building rules and models for troubleshooting
  - automation shifts the error source from operator errors to design errors
    - » harder to detect/tolerate/fix design errors

ACME: Learning from other fields: disasters



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 left in the wrong position; parts of a redundant system used only in an emergency.
   Facility running under normal operation masks errors in error handling

Recovery-Oriented Computing

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

Summary: the present

- **After 15 years of working on performance, we need new and relevant goals**
  - **ACME:** Availability, Change, Maintainability, Evolutionary growth
- **Challenges in achieving ACME:**
  - Software in Internet services evolves rapidly
  - Hardware and software failures are inevitable
  - Human operator errors are inevitable
    - » Automation Irony tells us that we can't eliminate human
  - Test the emergency systems, remove latent errors
  - Traditional high-availability/fault-tolerance techniques don't solve the problem

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
- **The future:** Recovery-Oriented Computing (ROC)
- **ROC techniques and principles**

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*

- **Failures are a fact, and recovery/repair is how we cope with them**
- **Improving recovery/repair improves availability**
  - UnAvailability = $\frac{MTTR}{MTTF}$ *(assuming MTTR much less than MTTF)*
  - 1/10th MTTR just as valuable as 10X MTBF
- **Since major Sys Admin job is recovery after failure, ROC also helps with maintenance**
- **If necessary, start with clean slate, sacrifice disk space and performance for ACME**

Improving MTTR: approaches

- **Repair/recovery has 3 task components:**
  1) Detecting a problem
  2) Diagnosing the root cause of the problem
  3) Repairing the problem

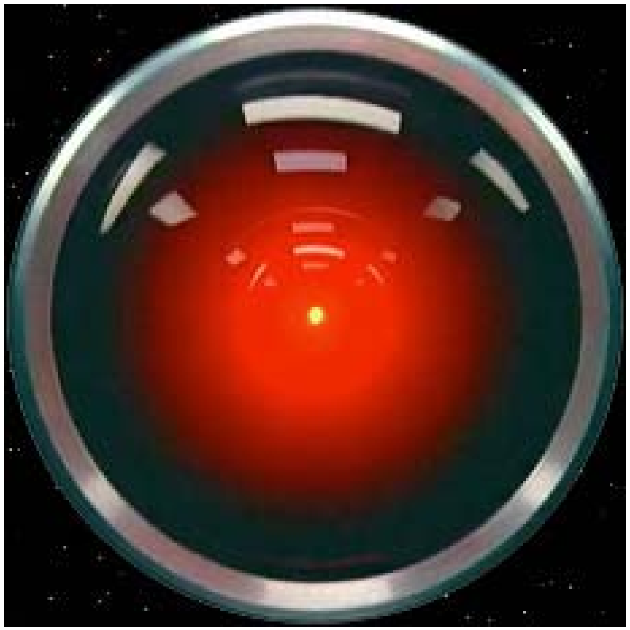
- **Two approaches to speeding up these tasks:**
  1) automate the entire process as a unit
     » the goal of most research into "self-healing", "self-maintaining", "self-tuning", or more recently "introspective" or "autonomic" systems
     see http://www.research.ibm.com/autonomic/

  2) ROC approach: provide tools to let human sysadmins carry out the three steps more effectively
     » if desired, add automation as a layer on top of the tools

Recovery-Oriented Computing

A science fiction analogy

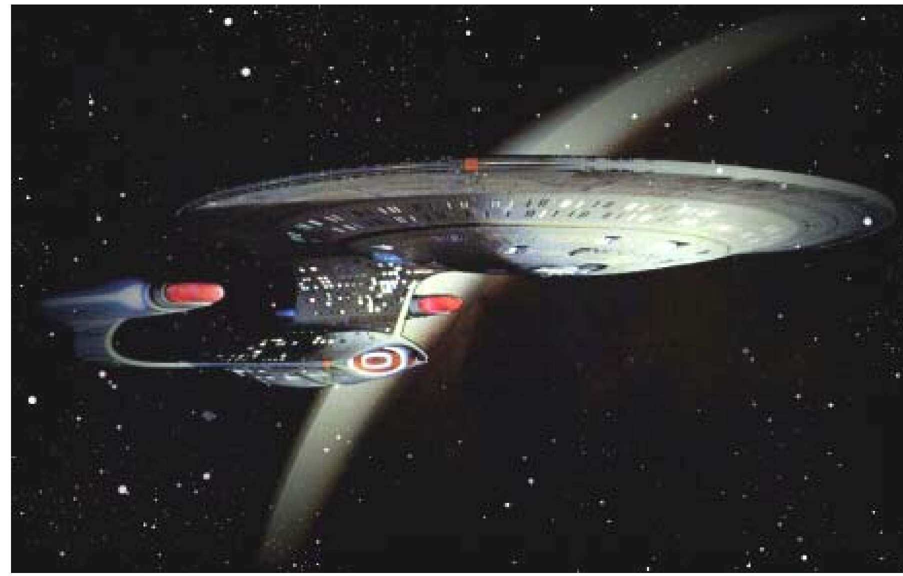
- **Autonomic approach**



HAL 9000 (2001)

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

- **ROC approach**



Enterprise computer (2365)

- **24th-century engineer is like today's sysadmin**
  - a *human* diagnoses & repairs computer problems
  - aided by diagnostic tools and understanding of system

Building human-aware recovery tools

- **Provide a safe, forgiving space 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

The Key to Human-Aware Recovery: Repairing the Past

- **Major goal of ROC is to provide an Undo for system administration**
  - to create an environment that forgives operator error
  - to let sysadmins fix latent errors even after they're manifested
    - » this is no ordinary word processor undo!
- **The Three R's: undo meets time travel**
  - **Rewind**: roll system state backwards in time
  - **Repair**: fix latent or active error
    - » automatically or via human intervention
  - **Redo**: roll system state forward, replaying user interactions lost during rewind

Recovery-Oriented Computing

Repairing the Past (2)

- **3 cases needing Undo**
  - reverse the effects of a mistyped command (rm -rf *)
  - roll back a software upgrade without losing user data
  - "go back in time" to retroactively install virus filter on email server; effects of virus are squashed on redo
- **The 3 R's vs. checkpointing, reboot, logging**
  - checkpointing gives Rewind only
  - reboot may give Repair, but only for "Heisenbugs"
  - logging can give all 3 R's
    - » but need more than RDBMS logging, since system state changes are interdependent and non-transactional
    - » 3R-logging requires careful dependency tracking, and attention to state granularity and externalized events

Recovery-Oriented Computing

Tools for Recovery #1: Detection

- **System enables input insertion, output check of all modules (<u>including fault insertion</u>)**
  - To check module sanity to find failures faster
  - To test correctness of recovery mechanisms
    - » insert (random) faults and known-incorrect inputs
    - » also enables availability benchmarks
  - To expose & remove latent errors from system
  - To train/expand experience of operator
    - » Periodic reports to management on skills
  - To discover if warning systems are broken

Tools for Recovery #2: 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

Recovery-Oriented Computing

ROC Enabler: isolation & 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
  - 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

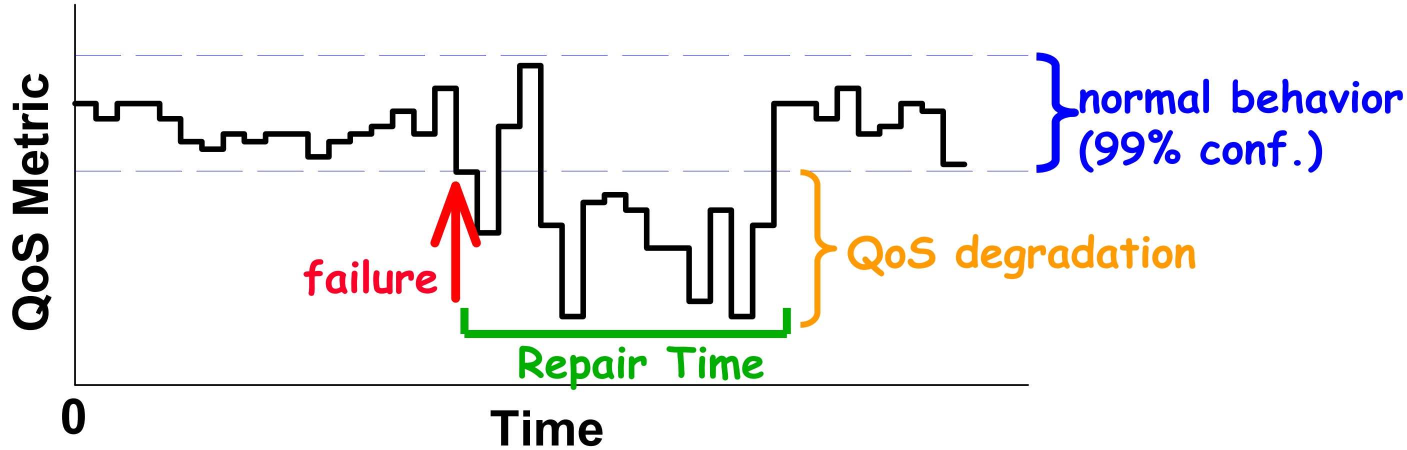
ROC Enabler: ACME benchmarks

- **Traditional benchmarks focus on performance**
  - ignore ACME goals
  - assume perfect hardware, software, human operators
- **New benchmarks needed to drive progress toward ACME, evaluate ROC success**
  - for example, *availability* and *recovery* benchmarks
  - How else convince developers, customers to adopt new technology?

RECOVERY-ORIENTED COMPUTING

Availability benchmarking 101

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

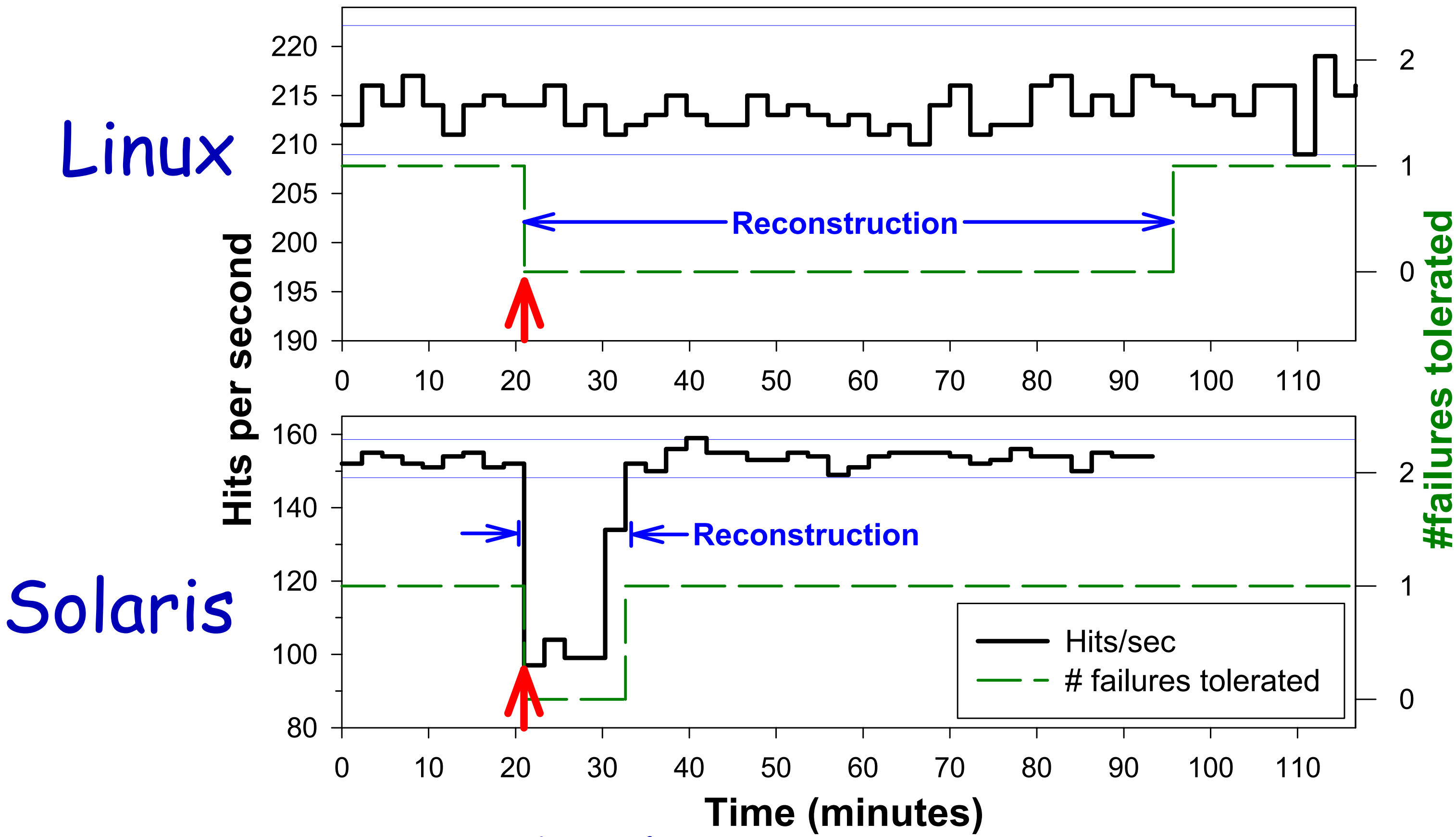


- **They require**
  - a realistic workload for the system
  - quality of service metrics and tools to measure them
  - fault-injection to simulate failures
  - human operators to perform repairs

Availability 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

Example: single-fault in SW RAID



- Compares Linux and Solaris reconstruction
  - **Linux**: minimal performance impact but longer window of vulnerability to second fault
  - **Solaris**: large perf. impact but restores redundancy fast
  - **Windows**: does not auto-reconstruct!

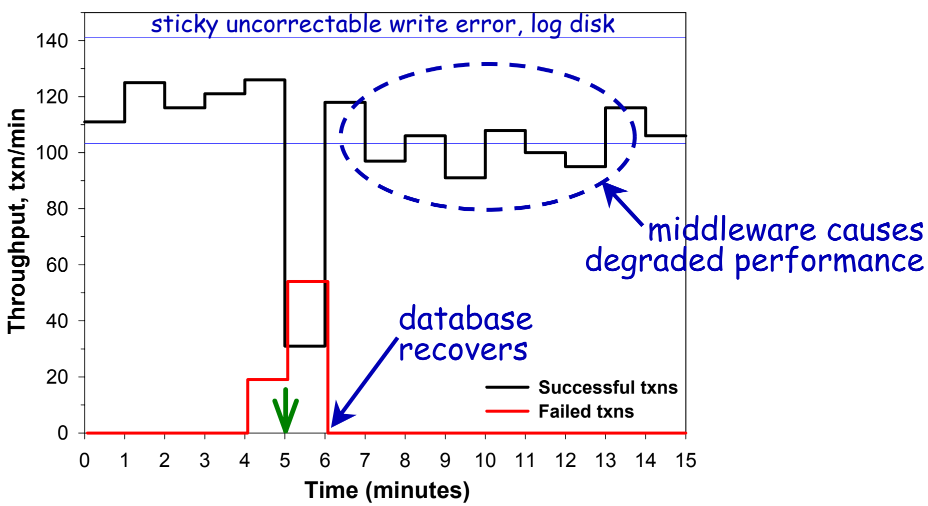
Software RAID: QoS behavior

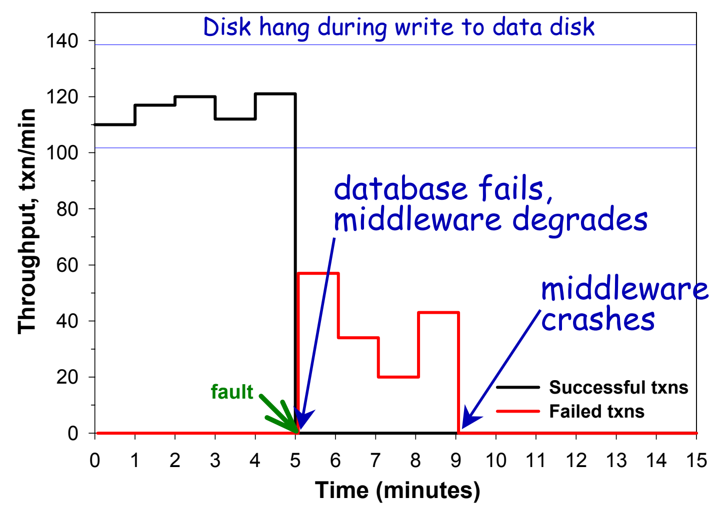
- **Response to double-fault scenario**
  - a double fault results in unrecoverable loss of data on the RAID volume
  - **Linux:** blocked access to volume
  - **Windows:** blocked access to volume
  - **Solaris:** silently continued using volume, delivering *fabricated* data to application!
    - » 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

Recovery-Oriented Computing

Example results: OLTP database

- **Setup**
  - 3-tier: Microsoft SQLServer/COM+/IIS & bus. logic
  - TPC-C-like workload; faults injected into DB data & log
- **Results**
  - Middleware highly unstable: degrades or crashes when DBMS fails or undergoes lengthy recovery





Recovery-Oriented Computing

Summary: from ROC to ACME

- **ROC: a new foundation to reduce MTTR**
  - Cope with fact that people, SW, HW fail (Peres's Law)
    - » the reality of fast-changing Internet services
  - Three R's to undo failures, bad repairs, fix the past
  - Human-focused designs to avoid Automation Irony and HAL-9000 effect, but still allow future automation
  - Self-verification to detect problems and latent errors
  - Diagnostics and root cause analysis to give ranking to potential solutions to problems
  - Recovery benchmarks to evaluate MTTR innovations
- **Significantly reducing MTTR (people/SW/HW)**
  => Significantly increased availability
  + Significantly improved maintenance costs

Interested in ROCing?

- Especially interested in collecting data on how real systems fail; let us know if you'd be willing to anonymously share data
- Also other ways for industrial participation
- See http://ROC.cs.berkeley.edu
- Contact Dave Patterson (patterson@cs.berkeley.edu) or Aaron Brown (abrown@cs.berkeley.edu)

BACKUP SLIDES

Evaluating ROC: human aspects

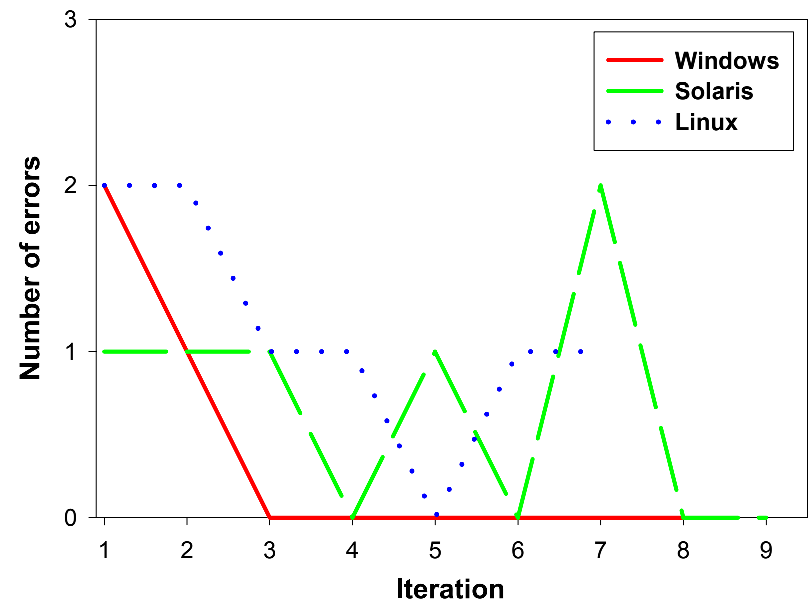
- **Must include humans in availability benchmarks**
  - to verify effectiveness of undo, training, diagnostics
  - humans act as system administrators
- **Subjects should be admin-savvy**
  - system administrators
  - CS graduate students
- **Challenge will be compressing timescale**
  - i.e., for evaluating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

Example results: software RAID (2)

- **Human error rates during repair**
  - 5 trained subjects repeatedly repairing disk failures

Error type	Windows	Solaris	Linux
Fatal Data Loss	💣		💣💣
Unsuccessful Repair			💣
System ignored fatal input			💣
User Error – Intervention Required	💣	💣💣	💣
User Error – User Recovered	💣	💣💣💣💣	💣💣
Total number of trials	**35**	**33**	**31**

  - errors rates do not decline with experience
    - » early: mistakes; later: slips & lapses
    - » UI has big impact on slips & lapses



Recovery-Oriented Computing