CS252
Graduate Computer Architecture

Lecture 6:
I/O 2: Failure Terminology, Examples, Gray Paper and a little Queueing Theory

February 2, 2001

Prof. David A. Patterson

Computer Science 252

Spring 2001

Review Storage

- Disks:
  - Extraodinary advance in capacity/drive, $/GB
  - Currently 17 Gbit/sq. in. ; can continue past 100 Gbit/sq. in.?
  - Bandwidth, seek time not keeping up: 3.5 inch form factor makes sense? 2.5 inch form factor in near future? 1.0 inch form factor in long term?
- Tapes
  - No investment, must be backwards compatible
  - Are they already dead?
  - What is a tapeless backup system?

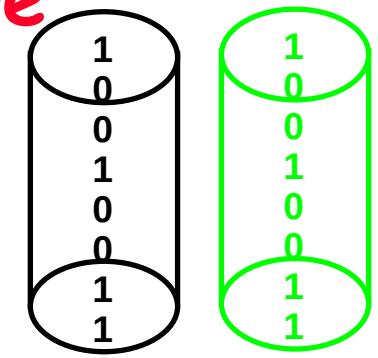
Review: RAID Techniques: Goal was performance, popularity due to reliability of storage

- *Disk Mirroring, Shadowing (RAID 1)*

  Each disk is fully duplicated onto its "shadow"

  Logical write = two physical writes

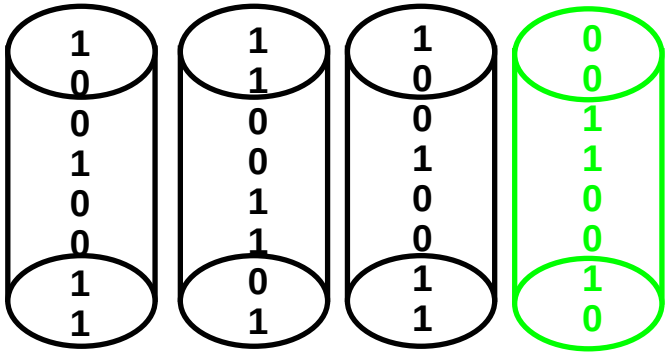
  100% capacity overhead



- *Parity Data Bandwidth Array (RAID 3)*

  Parity computed horizontally

  Logically a single high data bw disk

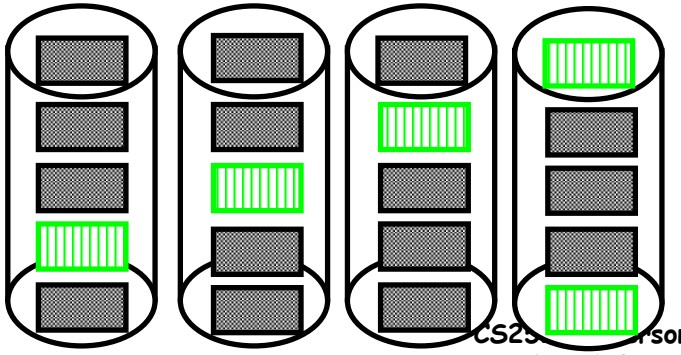


- *High I/O Rate Parity Array (RAID 5)*

  Interleaved parity blocks

  Independent reads and writes

  Logical write = 2 reads + 2 writes



Outline

- Reliability Terminology
- Examlpes
- Discuss Jim Gray's Turing paper

Definitions

- **Examples on why precise definitions so important for reliability**
- Is a programming mistake a fault, error, or failure?
  - Are we talking about the time it was designed or the time the program is run?
  - If the running program doesn't exercise the mistake, is it still a fault/error/failure?
- If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?
  - Is it a fault/error/failure if the memory doesn't access the changed bit?
  - Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?

IFIP Standard terminology

- Computer system *dependability*: quality of delivered service such that reliance can be placed on service
- *Service* is observed *actual behavior* as perceived by other system(s) interacting with this system's users
- Each module has ideal *specified behavior*, where *service specification* is agreed description of expected behavior
- A system *failure* occurs when the actual behavior deviates from the specified behavior
- failure occurred because an *error*, a defect in module
- The cause of an error is a *fault*
- When a fault occurs it creates a *latent error*, which becomes *effective* when it is activated
- When error actually affects the delivered service, a failure occurs (time from error to failure is *error latency*)

Fault v. (Latent) Error v. Failure

- A fault creates one or more latent errors
- Properties of errors are
  - a latent error becomes effective once activated
  - an error may cycle between its latent and effective states
  - an effective error often propagates from one component to another, thereby creating new errors
- Effective error is either a formerly-latent error in that component or it propagated from another error
- A component failure occurs when the error affects the delivered service
- These properties are recursive, and apply to any component in the system
- An error is manifestation *in the system* of a fault, a failure is manifestation *on the service* of an error

Fault v. (Latent) Error v. Failure

- An error is manifestation *in the system* of a fault, a failure is manifestation *on the service* of an error
- Is a programming mistake a fault, error, or failure?
  - Are we talking about the time it was designed or the time the program is run?
  - If the running program doesn't exercise the mistake, is it still a fault/error/failure?
- A programming mistake is a *fault*
- the consequence is an *error* (or *latent error*) in the software
- upon activation, the error becomes *effective*
- when this effective error produces erroneous data which affect the delivered service, a *failure* occurs

Fault v. (Latent) Error v. Failure

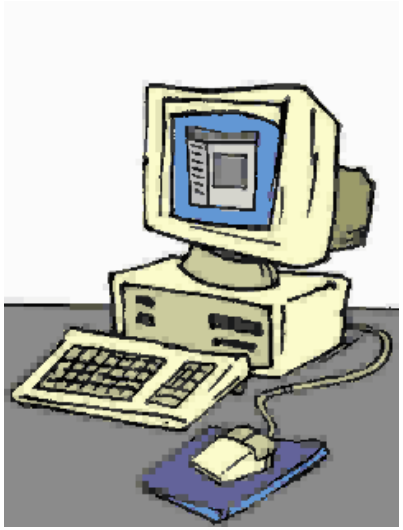
- An error is manifestation *in the system* of a fault, a failure is manifestation *on the service* of an error
- Is If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?
  - Is it a fault/error/failure if the memory doesn't access the changed bit?
  - Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?
- An alpha particle hitting a DRAM can be a fault
- if it changes the memory, it creates an error
- error remains latent until effected memory word is read
- if the effected word error affects the delivered service, a failure occurs

Fault v. (Latent) Error v. Failure

- An error is manifestation *in the system* of a fault, a failure is manifestation *on the service* of an error
- What if a person makes a mistake, data is altered, and service is affected?
- fault:
- error:
- latent:
- failure:

Fault Tolerance vs Disaster Tolerance

- Fault-Tolerance (or more properly, Error-Tolerance): **mask local faults (prevent errors from becoming failures)**
  - **RAID disks**
  - **Uninterruptible Power Supplies**
  - **Cluster Failover**
- Disaster Tolerance: **masks site errors (prevent site errors from causing service failures)**
  - **Protects against fire, flood, sabotage,..**
  - **Redundant system and service at remote site.**
  - **Use design diversity**



From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

CS 252 Administrivia

- Send 1-2 paragraph summary of papers to Yu-jia Jin (yujia@ic.eecs) BEFORE CLASS Wednesday
  - Hennessy, J. "The future of systems research."
  - Should have already turned in
    - » G. MOORE, "Cramming More Components onto Integrated Circuits"
    - » J. S. LIPTAY, "Structural Aspects of the System/360 Model 85, Part II: The Cache"
    - » J.GRAY, Turing Award Lecture: "What Next? A dozen remaining IT problems"
- Please fill out Third Edition chapter surveys for 6 by next Wednesday; 1,5 should be done
  - http://www.mkp.com/hp3e/quest-student.asp
- Project suggestions are on web site; start looking
  - http://www.cs.berkeley.edu/~pattrsn/252S01/suggestions.html
- Office hours Wednesdays 11-12

Defining reliability and availability quantitatively

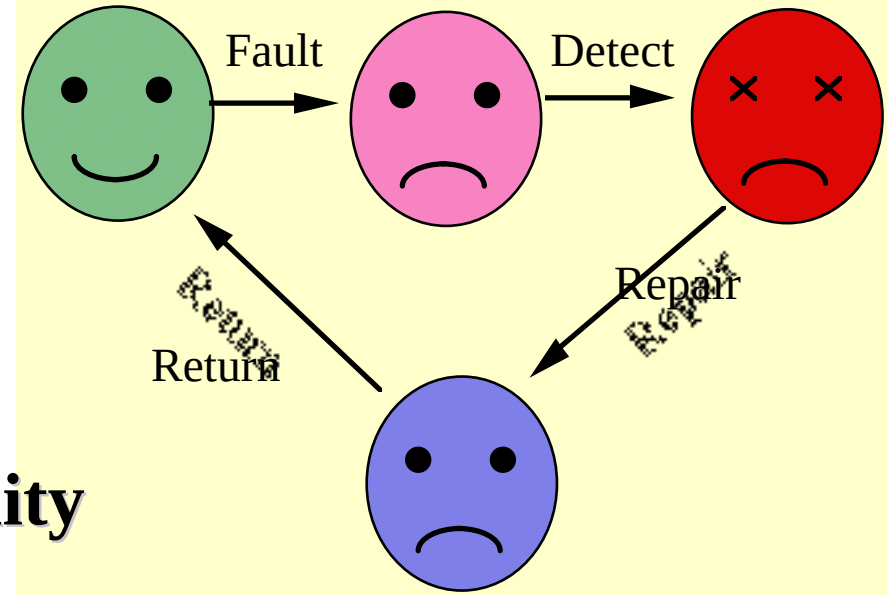
- Users perceive a system alternating between 2 states of service with respect to service specification:
  1. *service accomplishment*, where service is delivered as specified,
  2. *service interruption*, where the delivered service is different from the specified service, measured as Mean Time To Repair (MTTR)

  Transitions between these 2 states are caused by *failures* (from state 1 to state 2) or *restorations* (2 to 1)

- *module reliability:* a measure of continuous service accomplishment (or of time to failure) from a reference point, e.g, Mean Time To Failure (MTTF)
  - The reciprocal of MTTF is failure rate
- *module availability:* measure of service accomplishment with respect to alternation between the 2 states of accomplishment and interruption
  = MTTF / (MTTF+MTTR)

Fail-Fast is Good, Repair is Needed

Lifecycle of a module
fail-fast gives
short fault latency



High Availability
is low UN-Availability

Unavailability = $\frac{\text{MTTR}}{\text{MTTF+MTTR}}$

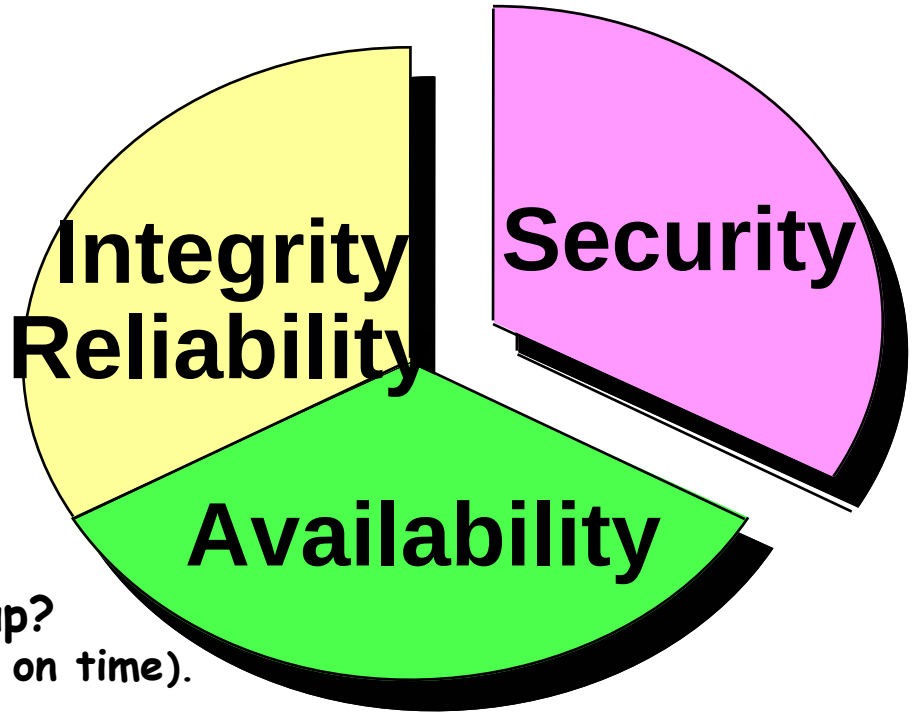
As MTTF>>MTTR, improving either MTTR or MTTF gives benefit

Note: Mean Time Between Failures (MTBF)= MTTF+MTTR

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Dependability: The 3 ITIES

- Reliability / Integrity: **does the right thing.**
  (Also large MTTF)
- Availability: **does it now.**
  (Also small $\frac{MTTR}{MTTF+MTTR}$
  System Availability:
  if 90% of terminals up & 99% of DB up?
  (=>89% of transactions are serviced on time).



From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Reliability Example

- If assume collection of modules have exponentially distributed lifetimes (age of compoent doesn't matter in failure probability) and modules fail independently, overall failure rate of collection is sum of failure rates of modules
- Calculate MTTF of a disk subsystem with
  - 10 disks, each rated at 1,000,000 hour MTTF
  - 1 SCSI controller, 500,000 hour MTTF
  - 1 power supply, 200,000 hour MTTF
  - 1 fan, 200,000 MTTF
  - 1 SCSI cable, 1,000,000 hour MTTF
- Failure Rate = 10*1/1,000,000 + 1/500,000 + 1/200,000 + 1/200,000 + 1/1,000,000 = (10 +2 +5 +5 +1)/1,000,000 = 23/1,000,000
- MTTF=1/Failure Rate = 1,000,000/23 = 43,500 hrs

What's wrong with MTTF?

- 1,000,000 MTTF > 100 years; ~ infinity?
- How calculated?
  - Put, say, 2000 in a room, calculate failures in 60 days, and then calculate the rate
  - As long as <=3 failures => 1,000,000 hr MTTF
- Suppose we did this with people?
- 1998 deaths per year in US ("Failure Rate")
- Deaths 5 to 14 year olds = 20/100,000
- $MTTF_{human}$ = 100,000/20 = 5,000 years
- Deaths >85 year olds = 20,000/100,000
- $MTTF_{human}$ = 100,000/20,000 = 5 years

source: "Deaths: Final Data for 1998," www.cdc.gov/nchs/data/nvs48_11.pdf

What's wrong with MTTF?

- 1,000,000 MTTF > 100 years; ~ infinity?
- But disk lifetime is 5 years!
- => if you replace a disk every 5 years, on average it wouldn't fail until 21st replacement
- A better unit: % that fail
- Fail over lifetime if had 1000 disks for 5 years
  = (1000 disks * 365*24) / 1,000,000 hrs/failure
  = 43,800,000 / 1,000,000 = 44 failures
  = 4.4% fail with 1,000,000 MTTF
- Detailed disk spec lists failures/million/month
- Typically about 800 failures per month per million disks at 1,000,000 MTTF,
  or about 1% per year for 5 year disk lifetime

Dependability Big Idea: No Single Point of Failure

- Since Hardware MTTF is often 100,000 to 1,000,000 hours and MTTF is often 1 to 10 hours, there is a good chance that if one component fails it will be repaired before a second component fails
- Hence design systems with sufficient redundancy that there is No Single Point of Failure

HW Failures in Real Systems: Tertiary Disks

- A cluster of 20 PCs in seven 7-foot high, 19-inch wide racks with 368 8.4 GB, 7200 RPM, 3.5-inch IBM disks. The PCs are P6-200MHz with 96 MB of DRAM each. They run FreeBSD 3.0 and the hosts are connected via switched 100 Mbit/second Ethernet

Component	Total in System	Total Failed	% Failed
SCSI Controller	44	1	2.3%
SCSI Cable	39	1	2.6%
SCSI Disk	368	7	1.9%
IDE Disk	24	6	25.0%
Disk Enclosure -Backplane	46	13	28.3%
Disk Enclosure - Power Supply	92	3	3.3%
Ethernet Controller	20	1	5.0%
Ethernet Switch	2	1	50.0%
Ethernet Cable	42	1	2.3%
CPU/Motherboard	20	0	0%

When To Repair?

Chances Of Tolerating A Fault are 1000:1 (class 3)

A 1995 study: Processor & Disc Rated At ~ 10khr MTTF

Computed Single Failures	Observed Double Fails	Ratio
10k Processor Fails	14 Double	~ 1000 : 1
40k Disc Fails,	26 Double	~ 1000 : 1

Hardware Maintenance:

On-Line Maintenance "Works" 999 Times Out Of 1000.

The chance a duplexed disc will fail during maintenance?1:1000

Risk Is 30x Higher During Maintenance

=> Do It Off Peak Hour

Software Maintenance:

Repair Only Virulent Bugs

Wait For Next Release To Fix Benign Bugs

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Sources of Failures

	MTTF	MTTR	
Power Failure:	2000 hr	1 hr	
Phone Lines			
Soft	>.1 hr	.1 hr	
Hard	4000 hr	10 hr	
Hardware Modules:	100,000hr	10hr	(many are transient)

Software:

1 Bug/1000 Lines Of Code (after vendor-user testing)

=> Thousands of bugs in System!

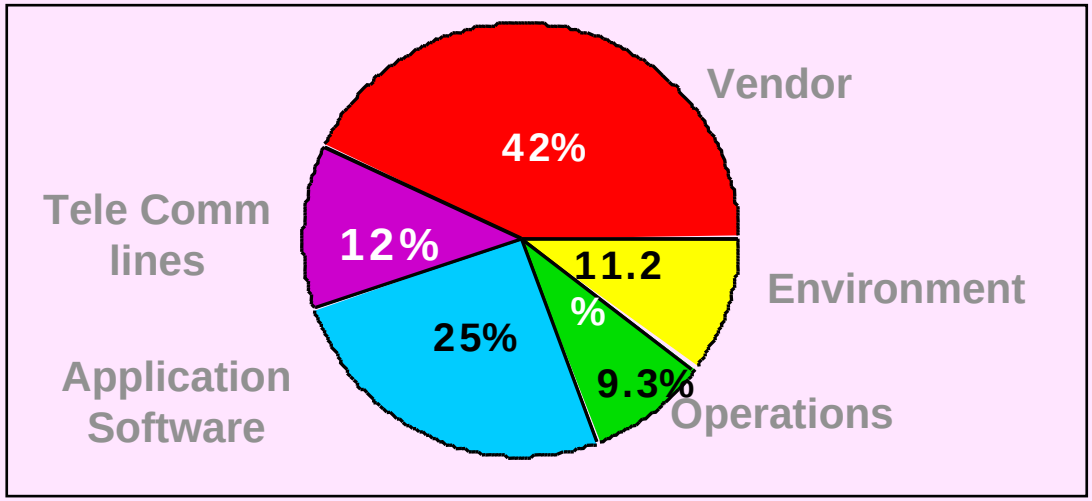
Most software failures are transient: dump & restart system.

Useful fact: 8,760 hrs/year ~ 10k hr/year

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

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

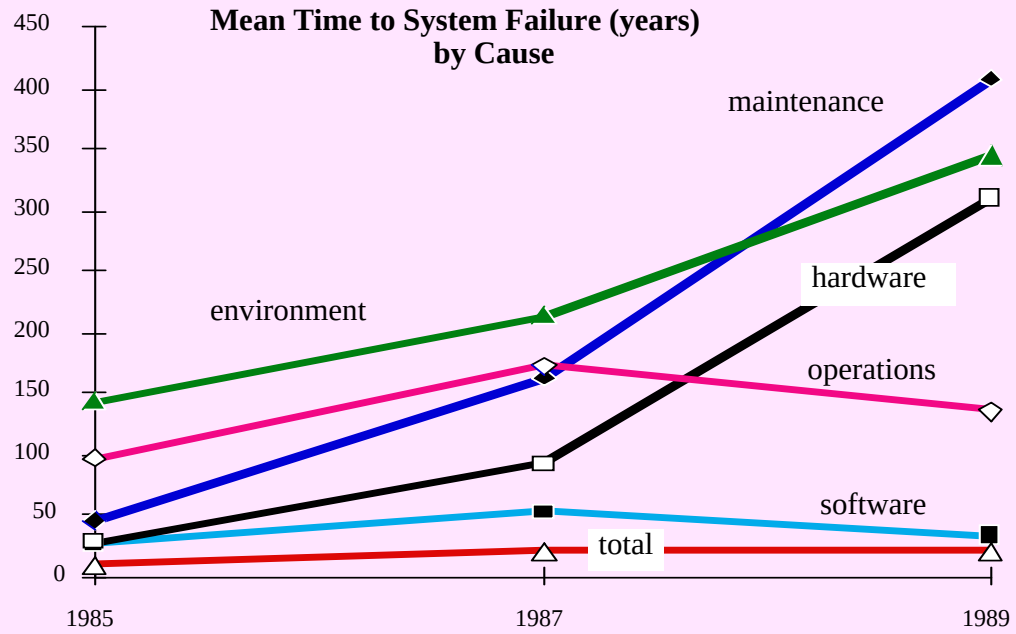
1,383 institutions reported (6/84 - 7/85)
7,517 outages, MTTF ~ 10 weeks, avg duration ~ 90 MINUTES

To Get 10 Year MTTF, Must Attack All These Areas

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Case Studies - Tandem Trends

Reported MTTF by Component



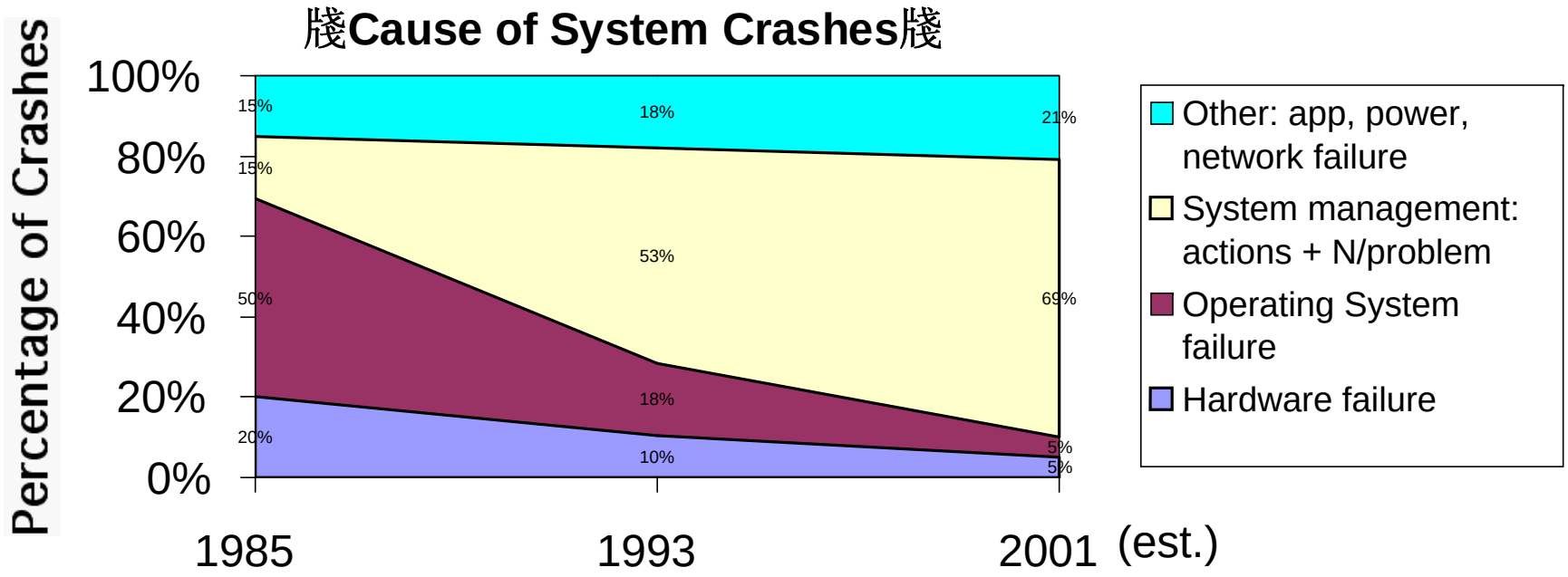
	1985	1987	1990	
SOFTWARE	2	53	33	Years
HARDWARE	29	91	310	Years
MAINTENANCE	45	162	409	Years
OPERATIONS	99	171	136	Years
ENVIRONMENT	142	214	346	Years
SYSTEM	8	20	21	Years

Problem: Systematic Under-reporting

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Is Maintenance the Key?

- **Rule of Thumb: Maintenance 10X HW**
  - **so over 5 year product life, ~ 95% of cost is maintenance**



- **VAX crashes '85, '93 [Murp95]; extrap. to '01**
- **Sys. Man.:** N crashes/problem, SysAdmin action
  - Actions: set params bad, bad config, bad app install
- **HW/OS 70% in '85 to 28% in '93. In '01, 10%?**

OK: So Far

Hardware fail-fast is easy

Redundancy plus Repair is great (Class 7 availability)

Hardware redundancy & repair is via modules.

How can we get instant software repair?

We Know How To Get Reliable Storage

RAID Or Dumps And Transaction Logs.

We Know How To Get Available Storage

Fail Soft Duplexed Discs (RAID 1...N).

? How do we get reliable execution?

? How do we get available execution?

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Does Hardware Fail Fast? 4 of 384 Disks that failed in Tertiary Disk

Messages in system log for failed disk	No. log msgs	Duration (hours)
Hardware Failure (Peripheral device write fault [for] Field Replaceable Unit)	1763	186
Not Ready (Diagnostic failure: ASCQ = Component ID [of] Field Replaceable Unit)	1460	90
Recovered Error (Failure Prediction Threshold Exceeded [for] Field Replaceable Unit)	1313	5
Recovered Error (Failure Prediction Threshold Exceeded [for] Field Replaceable Unit)	431	17

High Availability System Classes
Goal: Build Class 6 Systems

System Type	Unavailable (min/year)	Availability	Availability Class
Unmanaged	50,000	90.%	1
Managed	5,000	99.%	2
Well Managed	500	99.9%	3
Fault Tolerant	50	99.99%	4
High-Availability	5	99.999%	5
Very-High-Availability	.5	99.9999%	6
Ultra-Availability	.05	99.99999%	7

UnAvailability = MTTR/MTBF

can cut it in ½ by cutting MTTR ***or*** MTBF

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

How Realistic is "5 Nines"?

- HP claims HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee "in certain pre-defined, pre-tested customer environments"
  - Application faults?
  - Operator faults?
  - Environmental faults?
- Collocation sites (lots of computers in 1 building on Internet) have
  - 1 network outage per year (~1 day)
  - 1 power failure per year (~1 day)
- Microsoft Network unavailable recently for a day due to problem in Domain Name Server: if only outage per year, 99.7% or 2 Nines

Demo: looking at some nodes

- Look at http://uptime.netcraft.com/
- Internet Node availability:
  - 92% mean,
  - 97% median

  Darrell Long (UCSC)

  ftp://ftp.cse.ucsc.edu/pub/tr/
  - ucsc-crl-90-46.ps.Z "A Study of the Reliability of Internet Sites"
  - ucsc-crl-91-06.ps.Z "Estimating the Reliability of Hosts Using the Internet"
  - ucsc-crl-93-40.ps.Z "A Study of the Reliability of Hosts on the Internet"
  - ucsc-crl-95-16.ps.Z "A Longitudinal Survey of Internet Host Reliability"

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Discuss Gray's Paper

- "What Next? A dozen remaining IT problems," June 1999, MS-TR-99-50
- http://research.microsoft.com/~gray/papers/MS_TR_99_50_TuringTalk.pdf

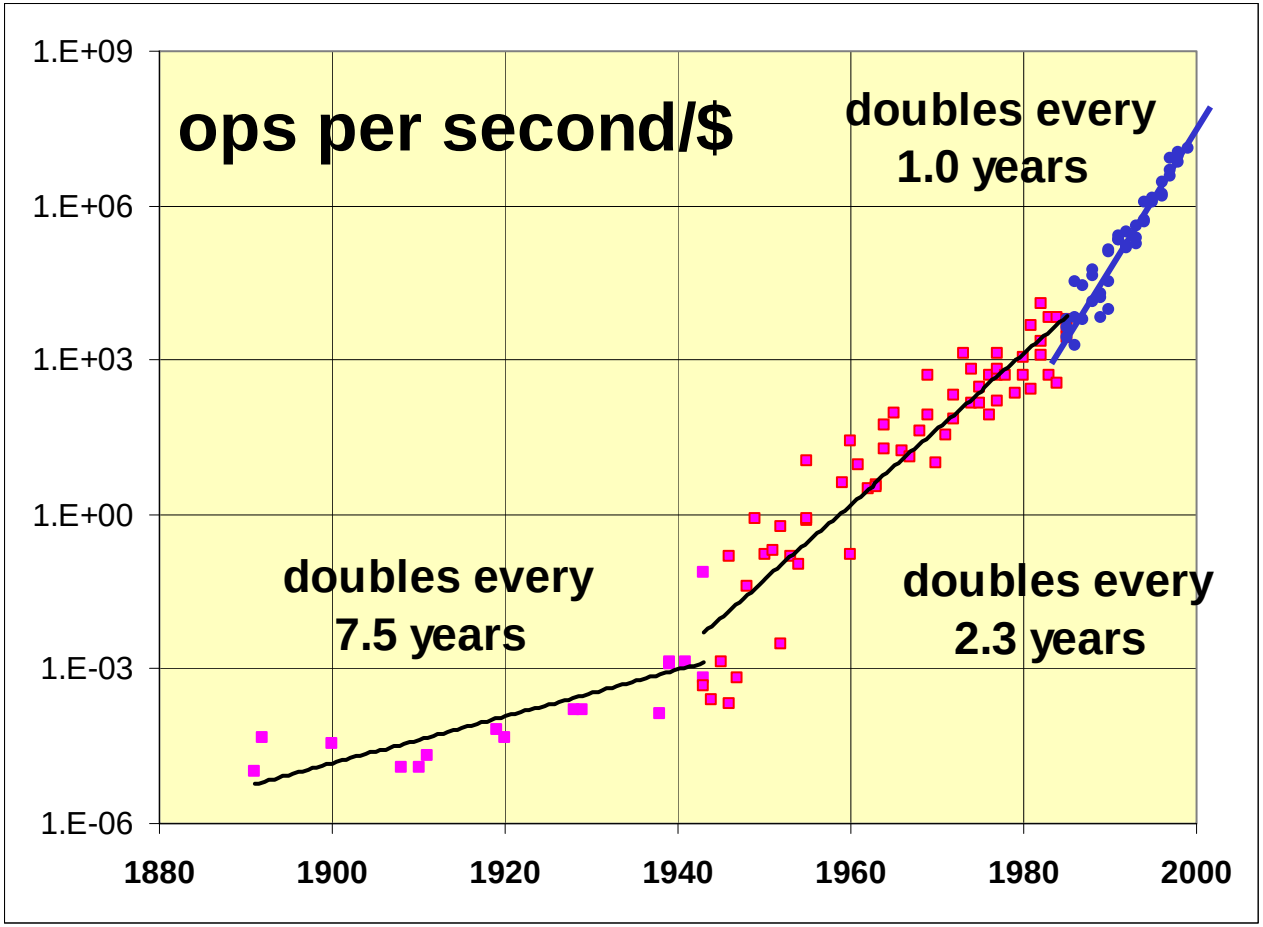
ops/s/$ Had Three Growth Curves 1890-1990

1890-1945
- Mechanical
- Relay
- 7-year doubling

1945-1985
- Tube, transistor,..
- 2.3 year doubling

1985-2000
- Microprocessor
- 1.0 year doubling

Combination of Hans Moravac + Larry Roberts + Gordon Bell
WordSize*ops/s/sysprice



The List (Red is AI Complete)

- Devise an architecture that scales up by 10^6.
- The Turing test: win the impersonation game 30% of the time.
  - 3.Read and understand as well as a human.
  - 4.Think and write as well as a human.
- Hear as well as a person (native speaker): speech to text.
- Speak as well as a person (native speaker): text to speech.
- See as well as a person (recognize).
  Illustrate as well as a person (done!) but virtual reality is still a major challenge.
- Remember what is seen and heard and quickly return it on request.
- Build a system that, given a text corpus, can answer questions about the text and summarize it as quickly and precisely as a human expert. Then add sounds: conversations, music. Then add images, pictures, art, movies.
- Simulate being some other place as an observer (Tele-Past) and a participant (Tele-Present).
- Build a system used by millions of people each day but administered by a ½ time person.
- Do 9 and prove it only services authorized users.
- Do 9 and prove it is almost always available: (out less than 1 second per 100 years).
- Automatic Programming: Given a specification, build a system that implements the spec. Prove that the implementation matches the spec. Do it better than a team of programmers.

Trouble-Free Systems

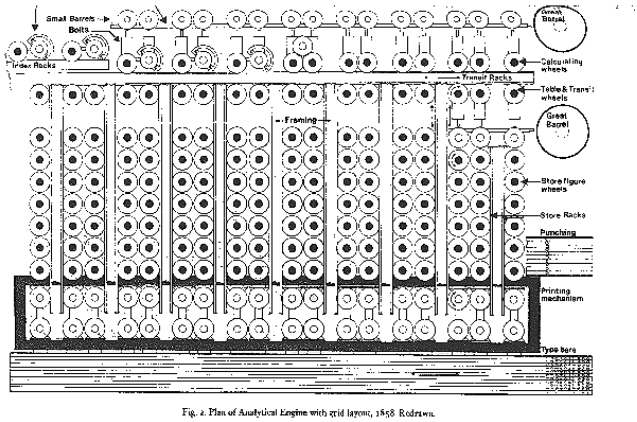


Fig. 2. Plan of Analytical Engine with grid layout, 1858 Redrawn.

- 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.

Trustworthy Systems

- Build a system used by millions of people that
  - Only services authorized users
    - » Service cannot be denied (can't destroy data or power).
    - » Information cannot be stolen.
  - Is always available: (out less than 1 second per 100 years = 8 9's of availability)
    - » 1950's 90% availability,
      Today 99% uptime for web sites,
      99.99% for well managed sites (50 minutes/year)
      3 extra 9s in 45 years.
    - » Goal: 5 more 9s: 1 second per century.
  - And prove it.

Summary: Dependability

- Fault => Latent errors in system => Failure in service
- Reliability: quantitative measure of time to failure (MTTF)
  - Assuming expoentially distributed independent failures, can calculate MTTF system from MTTF of components
- Availability: quantitative measure % of time delivering desired service
- Can improve Availability via greater MTTF or smaller MTTR (such as using standby spares)
- No single point of failure a good hardware guideline, as everything can fail
- Components often fail slowly
- Real systems: problems in maintenance, operation as well as hardware, software