EECS 252 Graduate Computer Architecture

Lec 20 – How to Have a Bad Career in Grad School and Beyond

David Patterson
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~pattrsn
http://vlsi.cs.berkeley.edu/cs252-s06

Outline

- Part I: Key Advice for a Bad Career while a Grad Student
- Part II: Advice on Alternatives to a Bad Graduate Career
- Administrivia
- Part III: Key Advice for a Bad Career
- Part IV: Key Advice on Alternatives to a Bad Career
- Topics covered in Parts III (today) and IV (next meeting)
  - Selecting a Problem
  - Picking a Solution
  - Performing the Research
  - Evaluating the Results
  - Communicating Results
  - Transferring Technology
- (Post-Tenure) Career Advice from Richard Hamming, Ivan Sutherland, Karl Pister (next meeting)
- Conclusion and Acknowledgments (next meeting)

Part I: How to Have a Bad Graduate Career

- Concentrate on getting good grades:
  - postpone research involvement: might lower GPA
- Minimize number and flavors of courses
  - Why take advantage of 1 of the top departments with an emphasis on excellent grad courses?
  - Why take advantage of a campus with 35/36 courses in the top 10?
  - May affect GPA
- Don’t trust your advisor
  - Advisor is only interested in his or her own career, not your’s
  - Advisor may try to mentor you, use up time, interfering with GPA
- Only work the number of hours per week you are paid!
  - Don’t let master class exploit the workers!
- Work at home as much as possible
  - Avoid distractions of talking to others

Part I: How to Have a Bad Graduate Career

- Concentrate on graduating as fast as possible
  - Winner is first in class to PhD
  - People only care about that you have a PhD and your GPA, not on what you know
    - » Nirvana: graduating in 3.5 years with a 4.0 GPA!
  - Don’t spend a summer in industry: takes longer
    - » How could industry experience help with selecting PhD topic?
  - Don’t work on large projects: takes longer
    - » Have to talk to others, have to learn different areas
    - » Synchronization overhead of multiple people
    - » You’ll get lost
  - Don’t do a systems PhD: takes longer
- Don’t go to conferences
  - It costs money and takes time; you’ll have plenty of time to learn the field after graduating
- Don’t waste time polishing writing or talks
  - Again, that takes time

Part II: Alternatives to a Bad Graduate Career

- Concentrate on getting good grades?
  - Reality: need to maintain reasonable grades
    - » Only once gave a below B in CS 252
    - » 3 prelim courses only real grades that count
  - What matters on graduation is letters of recommendation from 3-4 faculty/PhDs who have known you for 5+ years (including 1 outside of Berkeley: see summer jobs)
- Minimize number and flavors of courses?
  - Your last chance to be exposed to new ideas before have to learn them on your own (re: queueing theory and me)
  - Get a real outside minor from a campus with great departments in all fields; e.g., Management of Technology certificate, Copyright Law, SIM, Genetics, …
- Don't trust your advisor?
  - Primary attraction of campus vs. research lab is grad students
  - Faculty career is judged in large part by success of their students
  - Try taking advice of advisor?
- Stay home to avoid distractions?
  - Research is not a straight line; need to interact to get great ideas

Part II: Alternatives to a Bad Graduate Career

- Concentrate on graduating as fast as possible?
  - Your last chance to learn; most learning will be outside the classroom
  - Considered newly "minted" when finish PhD
    - » Judged on year of PhD vs. year of birth
    - » To a person in their 40s or 50s,
      1 or 2 more years is roundoff error (27 = 29)
- Don't go to conferences?
  - Chance to see firsthand what the field is like, where its going
  - There are student rates, you can share a room
  - Talk to people in the field in the halls
  - If your faculty advisor won't pay, then pay it yourself; almost always offer student rates, can often share rooms
    - » Landay paid his own way to conferences while grad student
- Don't waste time polishing writing or talks?
  - In the marketplace of ideas, the more polish the more likely people will pay attention to your ideas
  - Practice presentation AND answering tough questions

Part II: Alternatives to a Bad Graduate Career

- Only work the number of hours per week you are paid?
  - Campus Faculty average is 65-70 hours/work; EECS higher
  - Students should be in that range
  - Organize each day: when most alert? nap? exercise? sleep?
  - When/how often/how long: write, read, program, email?
  - To do lists: daily, weekly, semester
- Industrial Experience?
  - 1st or 2nd summer get work experience, or 1 semester off
- Advice from a very successful recent student; Remzi Arpaci (just got tenure at University Wisconsin – Madison)
  - Great ideas, did lots of papers, well thought of
  - I asked: Why do you think you did so well?
  - He said I gave him advice the first week he arrived
  - I asked: What did I say?
  - He said 3 observations, and still good advice today

Part II: How to be a Success in Graduate School

1) **"Swim or Sink"**

"Success is determined by me (student) primarily"

Faculty will set up the opportunity,
but its up to me leverage it

2) **"Read/learn on your own"**

"Related to 1), I think you told me this as you handed me a stack of about 20 papers"

3) **"Teach your advisor"**

"I really liked this concept; go out and learn about something and then teach the professor"

Fast moving field, don't expect prof to be
at forefront everywhere

CS252: Administrivia

- Still Grading Quiz 2
- Project Presentations Monday 5/1 in
  - Soda 6th Floor Alcove 9:45 to 4PM
- Signup sheet! Only 2 so far. Will assign randomly!
- Project Posters 5/3 Wednesday (11-12:30 in Soda 6th floor); lunch will be served
  - Cecilia has some poster boards in her office (676 Soda)...get them from her ASAP (she's on vacation Thursday-Monday). You will have to pick up easels for your posters on May 3rd before the poster session.
- Final Papers due Friday 5/5
  - Email Archana, who will post papers on class web site
- Last Meeting Monday 5/8: Goodbye to Computer Architecture / Your Cal Cultural History, Course Evaluation ; projects/grades handed out

Outline

- Part I: Key Advice for a Bad Career while a Grad Student
- Part II: Advice on Alternatives to a Bad Graduate Career
- Administrivia
- Part III: Key Advice for a Bad Career
- Part IV: Key Advice on Alternatives to a Bad Career
- Topics covered in Parts III (today) and IV (next meeting)
  - Selecting a Problem
  - Picking a Solution
  - Performing the Research
  - Evaluating the Results
  - Communicating Results
  - Transferring Technology
- (Post-Tenure) Career Advice from Richard Hamming, Ivan Sutherland, Karl Pister (next meeting)
- Conclusion and Acknowledgments (next meeting)

Bad Career Move #1: Be THE leading expert

- Invent a new field!
  - Make sure its slightly different
- Be the real Lone Ranger: Don't work with others
  - No ambiguity in credit
  - Adopt the Prima Donna personality
- Research Horizons
  - Never define success
  - Avoid Payoffs of less than 20 years
  - Stick to one topic for whole career
  - Even if technology appears to leave you behind, stand by your problem

Bad Career Move #2: Let Complexity Be Your Guide (Confuse Thine Enemies)

- Best compliment:
  "Its so complicated, I can't understand the ideas"
- Easier to claim credit for subsequent good ideas
  - If no one understands, how can they contradict your claim?
- It's easier to be complicated
  - Also: to publish it must be different; N+1st incremental change
- If it were not unsimple then how could distinguished colleagues in departments around the world be positively appreciative of both your extraordinary intellectual grasp of the nuances of issues as well as the depth of your contribution?

Bad Career Move #3: Never be Proven Wrong

- Avoid Implementing
- Avoid Quantitative Experiments
  - If you've got good intuition, who needs experiments?
  - Why give grist for critics' mill?
  - Plus, it takes too long to measure
- Avoid Benchmarks
- Projects whose payoff is ≥ 20 years gives you 19 safe years

Bad Career Move #4:
Use the Computer Scientific Method

Obsolete Scientific Method	Computer Scientific Method
• Hypothesis	• Hunch
• Sequence of experiment	• 1 experiment & change all parameters
• Change 1 parameter/ex	
• Prove/Disprove Hypothesis	• Discard if doesn't support hunch
• Document for others to reproduce results	• Why waste time? We know this

Bad Career Move #5:
Don't be Distracted by Others (Avoid Feedback)

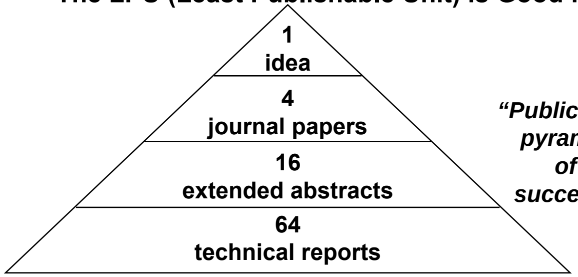
- Always dominate conversations:
  Silence is ignorance
  - Corollary: Loud is smart
- Don't read
- Don't be tainted by interaction with users, industry
- Reviews
  - If it's simple and obvious in retrospect ⇒ Reject
  - Quantitative results don't matter if they just show you what you already know ⇒ Reject
  - Everything else ⇒ Reject

Bad Career Move #6:
Publishing Journal Papers IS Technology Transfer

- Target Archival Journals: the Coin of the Academic Realm
  - It takes 2 to 3 years from submission to publication ⇒ timeless
- As the leading scientist, your job is to publish in journals;
  its *not* your job to make you the ideas palatable to the ordinary engineer
- Going to conferences and visiting companies just uses up valuable research time
  - Travel time, having to interact with others, serve on program committees, ...

Bad Career Move #7:
Writing Tactics for a Bad Career

- Papers: It’s Quantity, not Quality
  - Personal Success = Length of Publication List
  - “The LPU (Least Publishable Unit) is Good for You”



1
idea

4
journal papers

16
extended abstracts

64
technical reports

“Publication pyramid of success”

- Student productivity = number of papers
  - Number of students: big is beautiful
  - Never ask students to implement: reduces papers
- Legally change your name to Aaaanderson

5 Writing Commandments for a Bad Career

I. Thou shalt not define terms, nor explain anything.

II. Thou shalt replace “will do” with “have done”.

III. Thou shalt not mention drawbacks to your approach.

IV. Thou shalt not reference any papers.

V. Thou shalt publish before implementing.

7 Talk Commandments for a Bad Career

I. Thou shalt not illustrate.

II. Thou shalt not covet brevity.

III. Thou shalt not print large.

IV. Thou shalt not use color.

V. Thou shalt cover thy naked slides.

VI. Thou shalt not skip slides in a long talk.

VII. Thou shalt not practice.

Following all the commandments in Powerpoint!

- We describe the philosophy and design of the control flow machine, and present the results of detailed simulations of the performance of a single processing element. Each factor is compared with the measured performance of an advanced von Neumann computer running equivalent code. It is shown that the control flow processor compares favorably in the program.
- We present a denotational semantics for a logic program to construct a control flow for the logic program. The control flow is defined as an algebraic manipulator of idempotent substitutions and it virtually reflects the resolution deductions. We also present a bottom-up compilation of medium grain clusters from a fine grain control flow graph. We compare the basic block and the dependence sets algorithms that partition control flow graphs into clusters.
- A hierarchical macro-control-flow computation allows them to exploit the coarse grain parallelism inside a macrotask, such as a subroutine or a loop, hierarchically. We use a hierarchical definition of macrotasks, a parallelism extraction scheme among macrotasks defined inside an upper layer macrotask, and a scheduling scheme which assigns hierarchical macrotasks on hierarchical clusters.
- We apply a parallel simulation scheme to a real problem: the simulation of a control flow architecture, and we compare the performance of this simulator with that of a sequential one. Moreover, we investigate the effect of modeling the application on the performance of the simulator. Our study indicates that parallel simulation can reduce the execution time significantly if appropriate modeling is used.
- We have demonstrated that to achieve the best execution time for a control flow program, the number of nodes within the system and the type of mapping scheme used are particularly important. In addition, we observe that a large number of subsystem nodes allows more actors to be fired concurrently, but the communication overhead in passing control tokens to their destination nodes causes the overall execution time to increase substantially.
- The relationship between the mapping scheme employed and locality effect in a program are discussed. The mapping scheme employed has to exhibit a strong locality effect in order to allow efficient execution
- Medium grain execution can benefit from a higher output bandwidth of a processor and finally, a simple superscalar processor with an issue rate of ten is sufficient to exploit the internal parallelism of a cluster. Although the technique does not exhaustively detect all possible errors, it detects nontrivial errors with a worst-case complexity quadratic to the system size. It can be automated and applied to systems with arbitrary loops and nondeterminism.

7 Poster Commandments for a Bad Career

I. Thou shalt not illustrate.

II. Thou shalt not covet brevity.

III. Thou shalt not print large.

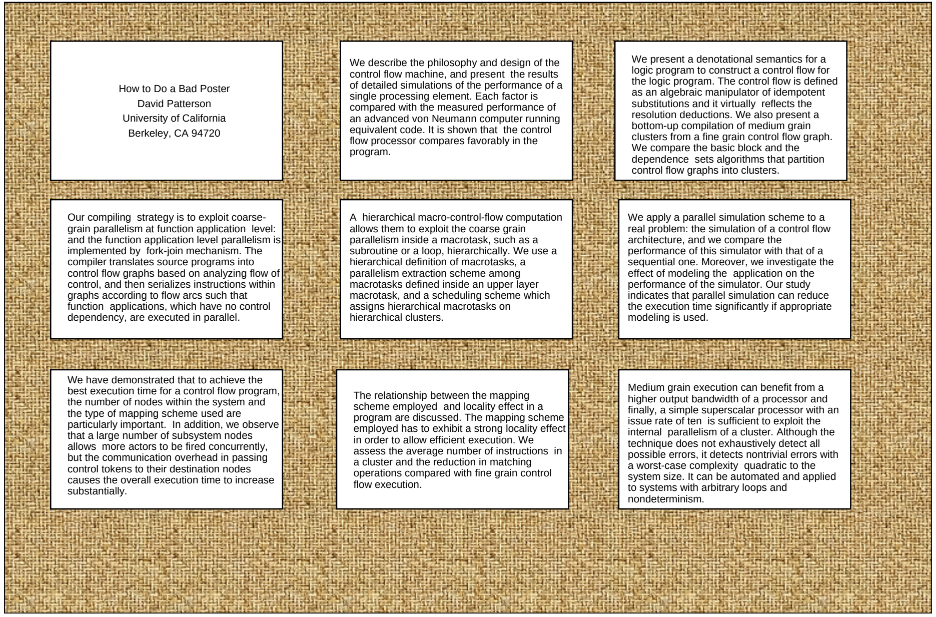
IV. Thou shalt not use color.

V. Thou shalt not attract attention to thyself.

VI. Thou shalt not prepare a short oral overview.

VII. Thou shalt not prepare in advance.

Following all the commandments



How to Do a Bad Poster
David Patterson
University of California
Berkeley, CA 94720

We describe the philosophy and design of the control flow machine, and present the results of detailed simulations of the performance of a single processing element. Each factor is compared with the measured performance of an advanced von Neumann computer running equivalent code. It is shown that the control flow processor compares favorably in the program.

We present a denotational semantics for a logic program to construct a control flow for the logic program. The control flow is defined as an algebraic manipulator of idempotent substitutions and it virtually reflects the resolution deductions. We also present a bottom-up compilation of medium grain clusters from a fine grain control flow graph. We compare the basic block and the dependence sets algorithms that partition control flow graphs into clusters.

Our compiling strategy is to exploit coarse-grain parallelism at function application level: and the function application level parallelism is implemented by fork-join mechanism. The compiler translates source programs into control flow graphs based on analyzing flow of control, and then serializes instructions within graphs according to flow arcs such that function applications, which have no control dependency, are executed in parallel.

A hierarchical macro-control-flow computation allows them to exploit the coarse grain parallelism inside a macrotask, such as a subroutine or a loop, hierarchically. We use a hierarchical definition of macrotasks, a parallelism extraction scheme among macrotasks defined inside an upper layer macrotask, and a scheduling scheme which assigns hierarchical macrotasks on hierarchical clusters.

We apply a parallel simulation scheme to a real problem: the simulation of a control flow architecture, and we compare the performance of this simulator with that of a sequential one. Moreover, we investigate the effect of modeling the application on the performance of the simulator. Our study indicates that parallel simulation can reduce the execution time significantly if appropriate modeling is used.

We have demonstrated that to achieve the best execution time for a control flow program, the number of nodes within the system and the type of mapping scheme used are particularly important. In addition, we observe that a large number of subsystem nodes allows more actors to be fired concurrently, but the communication overhead in passing control tokens to their destination nodes causes the overall execution time to increase substantially.

The relationship between the mapping scheme employed and locality effect in a program are discussed. The mapping scheme employed has to exhibit a strong locality effect in order to allow efficient execution. We assess the average number of instructions in a cluster and the reduction in matching operations compared with fine grain control flow execution.

Medium grain execution can benefit from a higher output bandwidth of a processor and finally, a simple superscalar processor with an issue rate of ten is sufficient to exploit the internal parallelism of a cluster. Although the technique does not exhaustively detect all possible errors, it detects nontrivial errors with a worst-case complexity quadratic to the system size. It can be automated and applied to systems with arbitrary loops and nondeterminism.

Outline

- Part I: Key Advice for a Bad Career while a Grad Student
- Part II: Advice on Alternatives to a Bad Graduate Career
- Administrivia
- Part III: Key Advice for a Bad Career
- Part IV: Key Advice on Alternatives to a Bad Career
- Topics covered in Parts III (today) and IV (next meeting)
  - Selecting a Problem
  - Picking a Solution
  - Performing the Research
  - Evaluating the Results
  - Communicating Results
  - Transferring Technology
- (Post-Tenure) Career Advice from Richard Hamming, Ivan Sutherland, Karl Pister (next meeting)
- Conclusion and Acknowledgments (next meeting)

Alternatives to Bad Papers

- Do opposite of Bad Paper commandments

  Define terms, distinguish "will do" vs "have done",

  mention drawbacks, real performance, reference other papers.
- Find related work via online search/paper retrieval (e.g., ACM portal) vs. www only

  `www.dbs.cdlib.org`
- First read Strunk and White, then follow these steps;

  1. 1-page paper outline, with tentative page budget/section
  2. Paragraph map
     - » 1 topic phrase/sentence per paragraph, handdrawn figures w. captions
  3. (Re)Write draft
     - » Long captions/figure can contain details ~ Scientific American
     - » Uses Tables to contain facts that make dreary prose
  4. Read aloud, spell check & grammar check
     (MS Word; Under Tools, select Grammar, select Options, select "technical" for writing style vs. "standard"; select Settings and select)
  5. Get feedback from friends and critics on draft; go to 3.
- `www.cs.berkeley.edu/~pattrsn/talks/writingtips.html`

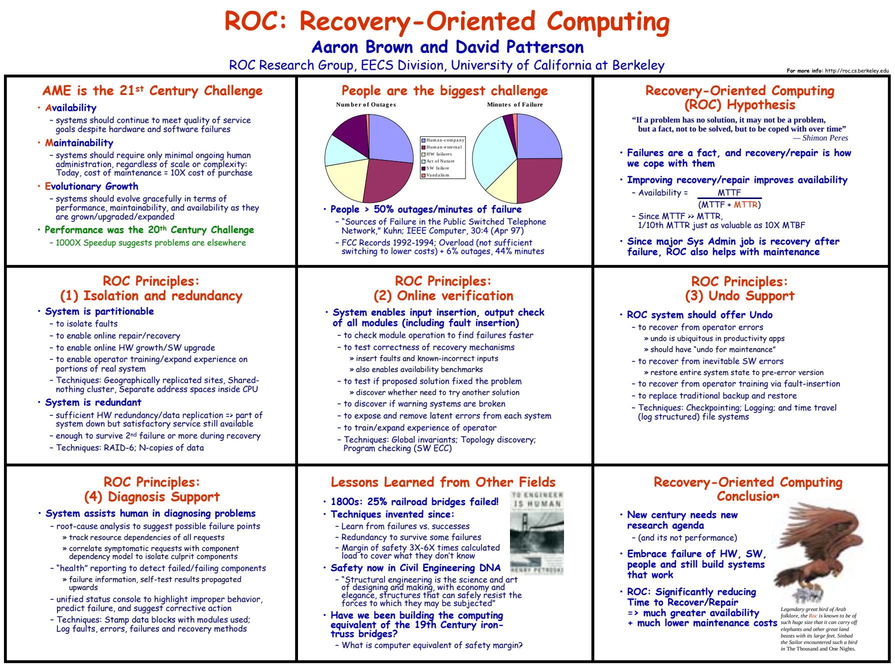
Alternatives to Bad Talks

- Do opposite of Bad Talk commandments
  - I. Thou shalt not illustrate.
  - II. Thou shalt not covet brevity.
  - III. Thou shalt not print large.
  - IV. Thou shalt not use color.
  - V. Thou shalt cover thy naked slides.
  - VI. Thou shalt not skip slides in a long talk.
  - VII. Thou shalt not practice.
- Allocate 1.5 to 2 minutes per slide (not counting title, outline sides), leave time for questions
- Don't over animate
- Do dry runs with friends/critics for feedback
  - – including tough audience questions
- Tape a practice talk (audio tape or video tape)
  - » Don't memorize speech, but have notes ready

Alternatives to Bad Posters (from Randy Katz)

- Answer Five Heilmeier Questions
  1. What is the problem you are tackling?
  2. What is the current state-of-the-art?
  3. What is your key make-a-difference concept or technology?
  4. What have you already accomplished?
  5. What is your plan for success?
- Do opposite of Bad Poster commandments
  - – Poster tries to catch the eye of person walking by
- 9 page poster might look like

Problem Statement	State-of-the-Art	Key Concept
Accomplish-ment # 1	Title and Visual logo	Accomplish-ment # 2
Accomplish-ment # 3	Plan for Success	Summary & Conclusion


Number of Outages
Minutes of Failure
Human-company
Human-external
HW failures
Act of Nature
SW failure
Vandalism
TO ENGINEER IS HUMAN
HENRY PETROSKI

ROC: Recovery-Oriented Computing

Aaron Brown and David Patterson

ROC Research Group, EECS Division, University of California at Berkeley

For more info: http://roc.cs.berkeley.edu

AME is the 21st Century Challenge

- Availability
  - – systems should continue to meet quality of service goals despite hardware and software failures
- Maintainability
  - – systems should require only minimal ongoing human administration, regardless of scale or complexity: Today, cost of maintenance = 10X cost of purchase
- Evolutionary Growth
  - – systems should evolve gracefully in terms of performance, maintainability, and availability as they are grown/upgraded/expanded
- Performance was the 20th Century Challenge
  - – 1000X Speedup suggests problems are elsewhere

People are the biggest challenge

- People > 50% outages/minutes of failure
  - – "Sources of Failure in the Public Switched Telephone Network," Kuhn; IEEE Computer, 30:4 (Apr 97)
  - – FCC Records 1992-1994; Overload (not sufficient switching to lower costs) + 6% outages, 44% minutes

Recovery-Oriented Computing (ROC) Hypothesis

"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
  - – Availability = $\frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})}$
  - – Since MTTF » MTTR, 1/10th MTTR just as valuable as 10X MTBF
- Since major Sys Admin job is recovery after failure, ROC also helps with maintenance

ROC Principles: (1) Isolation 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
  - – Techniques: Geographically replicated sites, Shared-nothing cluster, Separate address spaces inside CPU
- System is redundant
  - – sufficient HW redundancy/data replication => part of system down but satisfactory service still available
  - – enough to survive 2nd failure or more during recovery
  - – Techniques: RAID-6; N-copies of data

ROC Principles: (2) Online verification

- System enables input insertion, output check of all modules (including fault insertion)
  - – to check module operation to find failures faster
  - – to test correctness of recovery mechanisms
    - » insert faults and known-incorrect inputs
    - » also enables availability benchmarks
  - – to test if proposed solution fixed the problem
    - » discover whether need to try another solution
  - – to discover if warning systems are broken
  - – to expose and remove latent errors from each system
  - – to train/expand experience of operator
  - – Techniques: Global invariants; Topology discovery; Program checking (SW ECC)

ROC Principles: (3) Undo Support

- ROC system should offer Undo
  - – to recover from operator errors
    - » undo is ubiquitous in productivity apps
    - » should have "undo for maintenance"
  - – to recover from inevitable SW errors
    - » restore entire system state to pre-error version
  - – to recover from operator training via fault-insertion
  - – to replace traditional backup and restore
  - – Techniques: Checkpointing; Logging; and time travel (log structured) file systems

ROC Principles: (4) Diagnosis Support

- 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
  - – unified status console to highlight improper behavior, predict failure, and suggest corrective action
  - – Techniques: Stamp data blocks with modules used; Log faults, errors, failures and recovery methods

Lessons Learned from Other Fields

- 1800s: 25% railroad bridges failed!
- Techniques invented since:
  - – Learn from failures vs. successes
  - – Redundancy to survive some failures
  - – Margin of safety 3X-6X times calculated load to cover what they don't know
- Safety now in Civil Engineering DNA
  - – "Structural engineering is the science and art of designing and making, with economy and elegance, structures that can safely resist the forces to which they may be subjected"
- Have we been building the computing equivalent of the 19th Century iron-truss bridges?
  - – What is computer equivalent of safety margin?

Recovery-Oriented Computing Conclusion

- New century needs new research agenda
  - – (and its not performance)
- Embrace failure of HW, SW, people and still build systems that work
- ROC: Significantly reducing Time to Recover/Repair
  => much greater availability
  + much lower maintenance costs

Legendary great bird of Arab folklore, the Roc is known to be of such huge size that it can carry off elephants and other great land beasts with its large feet. Sinbad the Sailor encountered such a bird in The Thousand and One Nights.

Conclusions

- Part IV … to be continued next meeting