



How to Have a Bad Career as a Stanford Graduate Student

Christos Kozyrakis

<http://csl.stanford.edu/~christos>

September 2004



Outline

- Advice for a Bad Career while a Graduate Student
- Alternatives to a Bad Graduate Career
- Advice on How to Survive in a Large (Systems) Project
- If we have time
 - Advice for Bad Papers/Presentations/Posters
 - Alternatives to Bad Papers/Presentations/Papers
- Please interrupt me for questions & comments
 - This is not a lecture
 - There is not single correct approach to these issues

The History of this Talk

- A variation of Dave Patterson's talk on "How to Have a Bad Career in Industry/Academia"
 - Initial version in 1994, targeted to junior faculty
 - <http://www.cs.berkeley.edu/~pattrsn/talks/nontech.html>
- Modifications
 - Got rid of advice for junior faculty (too early for you)
 - Added some advice of my own
 - » Stanford-specific & big-project advice
- Who is Dave?
 - A computer science professor at Berkeley ☹
 - Research: RISC (Sparc), RAID, NOW (Clusters), IRAM, ROC
 - Research & Teaching awards from Berkeley, ACM, IEEE, ...
 - 2 seminal books on computer architecture (with John Hennessy)



Who Am I?

- Assistant professor of EE & CS
 - I teach EE282 (Computer Systems Architecture)
 - Research on architecture, compilation, programming models for parallel systems...
 - Stop by 304 Gates, visit <http://csl.stanford.edu/~christos>, or come to my CS300 talk (10/5, 4.15pm) for more info
- My graduate student background
 - CS Ph.D. at Berkeley (2002) ☹
 - » Major: Computer Systems
 - » Minors: Digital Circuits, Management of Technology (Business)
 - Undergrad abroad (U. of Crete, Greece)
 - Part of a large systems project (IRAM) that implemented prototypes
 - » 2-3 professors, many students, multiple companies
 - » Worked on architecture, design, simulators, and benchmarking
 - Took me 6 years...
 - » ... but I got a job offer 1 year before filling my thesis ☺



Part I:

**How to Have a Bad Graduate
Career**



How to Have a Bad Graduate Career (1)

- Concentrate on getting good grades
 - postpone research involvement: might lower GPA
- Minimize number and flavors of courses (CS students)
 - Why take advantage of one of the top departments with a wide range of excellent grad courses
 - Why take advantage of one of the best universities in the world?
 - May affect GPA
- Don't trust your advisor
 - Advisor is only interested in his or her own career, not yours
 - Advisor may try to mentor you, use up time, interfering with GPA
- Only work the number of hours per week you are paid!
 - Even less if possible...
 - Don't let master class exploit the workers!

How to Have a Bad Graduate Career (2)

- Concentrate on graduating as fast as possible
 - Winner is first in class to Ph.D.
 - People only care about that you have a Ph.D. 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 Ph.D. topic?
 - Don't work on large projects: takes longer
 - » Have to talk to others, have to learn different areas
 - » Synchronization overhead of multiple people
 - Don't do a systems Ph.D.: 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

How to Have a Bad Graduate Career (3)

- **Worry too much about the Qualls (EE students)**
 - Their only purpose is to make you look stupid & fail
 - They also take time, lower GPA, delay graduation, ...
 - Make it the goal of your life for the first couple of years
 - Don't take them your first year, more likely you will fail...
- **Never read on your own**
 - Takes time, ...
 - What's the advisor for if you have to pick your own readings?
 - May lead to extra work, interaction with others, ...
- **Work in the specific area you applied for**
 - E.g. if you applied for networking, stay in networking no matter what
 - Your application is a binding contract
 - Leverage your previous experience, graduate faster!

How to Have a Bad Graduate Career (4)

- Don't worry at all about research funding
 - It's your advisor's job to keep you "in the money"
 - Why waste any time with proposals, fellowship applications, etc?
 - Fellowships just make it more difficult to write a 2-page CV...
- Worry a lot about research funding
 - If you don't have a fellowship, you are a 2nd class citizen
 - If you are asked to TA, you are a 3rd class citizen



Part II:

Alternatives to a Bad Graduate Career



Alternatives to a Bad Graduate Career (1)

- Concentrate on getting good grades?
 - Reality: need to maintain reasonable grades
 - » For minimum school/department requirements
 - » Do PhD students ever get a B?
 - What matters on graduation is letters of recommendation from 3-4 faculty/Ph.D.s who have known you for 5+ years
- Minimize number and flavors of courses (CS students)?
 - Your last chance to be exposed to new ideas before have to learn them on your own (re: compilers and me)
 - Search for interesting courses beyond your area/department
 - » EE↔CS, Bioengineering, IP/cyber law, management of technology, etc
- Don't trust your advisor?
 - Primary attraction of campus vs. research lab is getting to work with grad students
 - Faculty career is judged in large part by success of his or her students
 - Try taking advice of advisor!

Alternatives to a Bad Graduate Career (2)

- Concentrate on graduating as fast as possible?
 - Your last chance to learn; most learning will be outside the classroom
 - Considered newly "minted" when finish Ph.D.
 - » Judged on year of Ph.D. vs. year of birth
 - » To a person in their 40s/50s, 1-2 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
- 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

Alternatives to a Bad Graduate Career (3)

- Only work the number of hours per week you are paid?
 - Faculty average is 65-70 hours/week
 - 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
 - Keep good notes of your ideas, problems, ...
 - Work hard, play hard
- Industrial Experience?
 - 1st or 2nd summer get work experience, or 1 semester off
- Worry about Qualls?
 - If we did not think you are smart enough, you would not be here
 - Designed to improve breadth
 - See it as an interesting challenge, not as an obstacle
 - Take it your first year, pass it, and forget about it...

Alternatives to a Bad Graduate Career (4)

- Never switch field?
 - Technology changes, opportunities change, interests change...
 - Explore opportunities across the department
 - Use classes and class projects to experiment with other fields
 - It gets more difficult to switch fields later on...
- Don't participate in a large project?
 - Do it, if you find an interesting project
 - See later part of the talk
- Funding
 - Help your advisor secure funding
 - » Learning how to present research to sponsors is a useful skill
 - » A fellowship in your CV looks very good so apply for one if you can...
 - RA/TAs are also good
 - » At the end, your research is what matters
 - » If teaching was not important, your advisor would not be here...

The Highlights

1. "Swim or Sink"
 - Success is determined primarily by the student
 - Faculty set up opportunity, but up to you to leverage it
 - Show initiative! Show initiative! Show initiative!
 - Selecting your PhD topic is as important as your results
2. "Read/learn on your own"
 - Fast moving field, don't expect Prof to be at forefront everywhere
 - Read papers, got to seminars/conferences, talk to colleagues
 - Once you know about something, teach your advisor...
3. "Ask Questions"
 - Lots of smart people in grad school (and even on the faculty), but don't be intimidated.
 - Either they know and you will learn, or they don't know and you will all learn by trying to determine the answer
 - It's OK to ask a "stupid question" every now and then
4. "Be honest about your work"
 - Your reputation as a researcher is valuable asset
 - Don't promise something you know you cannot deliver
 - Be honest about your results and progress
 - Long term, your colleagues or advisor will find out about it

Choosing an Advisor

- Very important decision , don't have to rush into it
- Tips
 - Get to know your potential advisor
 - » Talk to advisor, her students, attend their group meetings
 - A good advisor for is the one
 - » Has the right project or works on right area
 - » Has time for you (not too many students)
 - » Has the right research style
 - » Can provide you with funding (sooner or later)
 - » Your characters are compatible
 - Shop around
 - You can start on a trial basis
 - » Class or independent projects are a good way to do this
- Remember: you can switch advisors, but it's usually expensive...

Working with your advisor

- Your advisor is your academic parent
 - Most of what you learned growing up applies
- Tips
 - Talk with your advisor
 - » What you are working on and how, needs and worries (short & long term)
 - Don't be afraid to disagree with them
 - If you keep your advisor happy, she will keep you happy!
 - » For better or worse, you depend on each other 😊
- Watch out for
 - Advisors are also humans
 - » They can make mistakes or be harsh, especially under lots of pressure
 - Computer engineers/scientists are not known for their social skill 😞
 - Don't lose the respect/trust of your advisor!

Remember that...



Picking a research topic

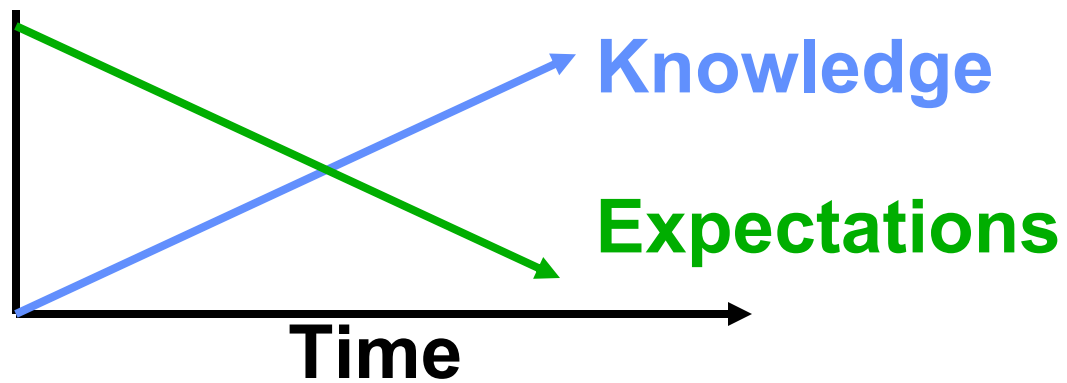
- Selecting the problem is as important as the research on it
- You don't have to select it now
 - Get started with something fun soon!
 - Good research training (techniques, important problems etc)
 - Thesis topic will be clear in ~2-3 years (hopefully)
 - » Don't be surprised if it is very different that initial plan...
- You have to like your research topic
 - Only way to keep sane during the "dark periods"
- Make sure that your problem has intermediate milestones
 - Good for publishing papers
 - Good for checking progress & feasibility

From research to publishing in few easy steps



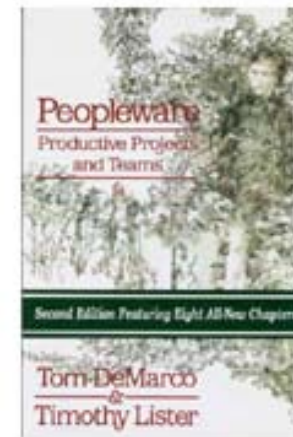
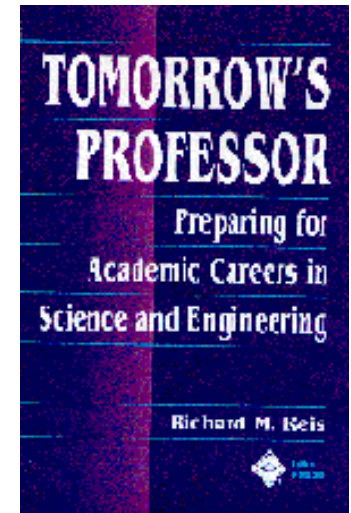
www.phdcomics.com


When to Graduate?



Some Good Reading


- "Technology And Courage", Ivan Sutherland (SUN)
 - http://research.sun.com/techrep/Perspectives/smli_ps-1.pdf
 - "Be bold; Take chances on hard topics"
- "You & Your Research", Richard Hamming (Bell Labs)
 - Richard Hamming (Bell Labs)
 - "Why do so few researchers make significant contributions?"
 - <http://www.cs.wisc.edu/~remzi/Postscript/hamming.ps>
- "The Task of a Referee", A. Smith
 - How to read & evaluate research papers
- "Tomorrow's Professors", Richard Reis (Stanford)
 - Great advice on Grad School & Future Career
 - Not limited to academics
- "Peopleware", Demarco & Lister
 - On teams, projects, & management
 - "Is your advisor a bad manager?"
 - Very entertaining...





Part III:

How to Survive a Large (Systems) Project



Advantages of Large Projects

- Get exposed to interdisciplinary research
 - Multiple fields: architecture, design, compilation, etc
 - You become an expert in a couple and knowledgeable in 3-4
- You develop a working system/prototype
 - Larger and faster impact, fame
 - Learn what it take to get something to really work
- Intellectual stimulation from more people
 - Faculty and students
- Learn to work in a team
 - That's how the "real world" works
 - Make life-long friends & partners
- Both industry and universities look for these qualities when they hire

(Assumed) Disadvantages of Large Projects

- Long implementation period
- Overhead of group interaction
- Identifying yourself within a large group

My opinion and experience:

- The benefits greatly outweigh the disadvantages
- Do it, if you find a project that you like
- Key to surviving a large project:
 - Show initiative
 - Keep a positive attitude and be flexible
 - Understand the stages the project will go through
 - » You are neither the first, nor the last one to go through this
 - » It all works out in the end

1. The "BrainStorming" Stage

- What's going on
 - All get to contribute to shaping the "new idea"
 - "New idea" papers based on initial studies on potential
 - Hot topic of discussion at conferences
 - A lot of excitement
- How you feel
 - "We will change the world"
 - "My famous advisor listens to me"
 - "I got a paper"
- Problems
 - None
- My advice
 - Enjoy it!
 - Prepare for the next phase

2. The "Implementation" Stage

- What's going on
 - Start building the prototype/system/infrastructure
 - Everybody in the group has an assigned task now
 - Start realizing the practical issues/limitations of the ideas
 - Few papers (if any)
 - Some newer project is the hot topic at the conferences/department
- How you feel:
 - "This is not research"
 - "Not a single publication for 2 years now"
 - "There is only 1 thesis in this project for 10 students!"
 - "I wish I was doing an individual project"
- My advise
 - Patience
 - Keep notes of what is difficult, could do better, want to change

2b. "The Stretch" or "The Really Dark Stage"

- What's going on
 - Final debugging before tape-out or software release, or the first period of operating the system
 - A huge amount of work under the pressure of deadlines
- How you feel
 - (same as in implementation stage)
 - "How long has it been since the last time I left the lab?"
 - "I just want to make it work and never see it again in my life"
 - "After this, my advisor will ask me to go find a PhD topic!"
- My advise
 - Hold on

3. When it All Comes Together

- Characteristics

- Working prototype/system
- Recognition of accomplishment by colleagues
- Tons of data, a lot of interesting answers, many papers
- Tons of data, many interesting observations on current/future implementations of various aspects of the system, many new questions and approaches, many theses
- Individual students follow their own way for the last year(s)

- How you feel

- "I got a thesis; I will graduate; Everybody wants to hire me"
- "I know how to make things work"
- (Looking at the prototype) "My baby!!!!"
- "I don't have enough time to explore all the new ideas/questions"
- "I got some great friends"



Part IV:

How to Prepare Bad Papers/Presentations/Posters



Writing Commandments for a Bad Career

1. Always make it sound complicated
2. Never define terms, never explain anything
3. Always replace "will do" with "have done"
 - Always publish before you implement
4. Never mention drawbacks of your approach
5. Never reference any papers (besides your own)
6. Never pay attention to the reviewers' comments

In case you did not receive the memo...

DECIPHERING ACADEMESE

YES, ACADEMIC LANGUAGE CAN BE OBTUSE, ABSTRUSE AND DOWNRIGHT DAEDAL. FOR YOUR CONVENIENCE, WE PRESENT A SHORT THESAURUS OF COMMON ACADEMIC PHRASES

"To the best of the author's knowledge..."

=

"WE WERE TOO LAZY TO DO A REAL LITERATURE SEARCH."

"It should be noted that..."

=

"OK, SO MY EXPERIMENTS WERENT PERFECT. ARE YOU HAPPY NOW??"

"Results were found through direct experimentation."

=

"WE PLAYED AROUND WITH IT UNTIL IT WORKED."

"These results suggest that..."

=

"IF WE TAKE A HUGE LEAP IN REASONING, WE CAN GET MORE MILEAGE OUT OF OUR DATA..."

"The data agreed quite well with the predicted model."

=

"IF YOU TURN THE PAGE UPSIDE DOWN AND SQUINT, IT DOESNT LOOK TOO DIFFERENT."

"Future work will focus on..."

=

"YES, WE KNOW THERE IS A BIG FLAW, BUT WE PROMISE WE'LL GET TO IT SOMEDAY."

"...remains an open question."

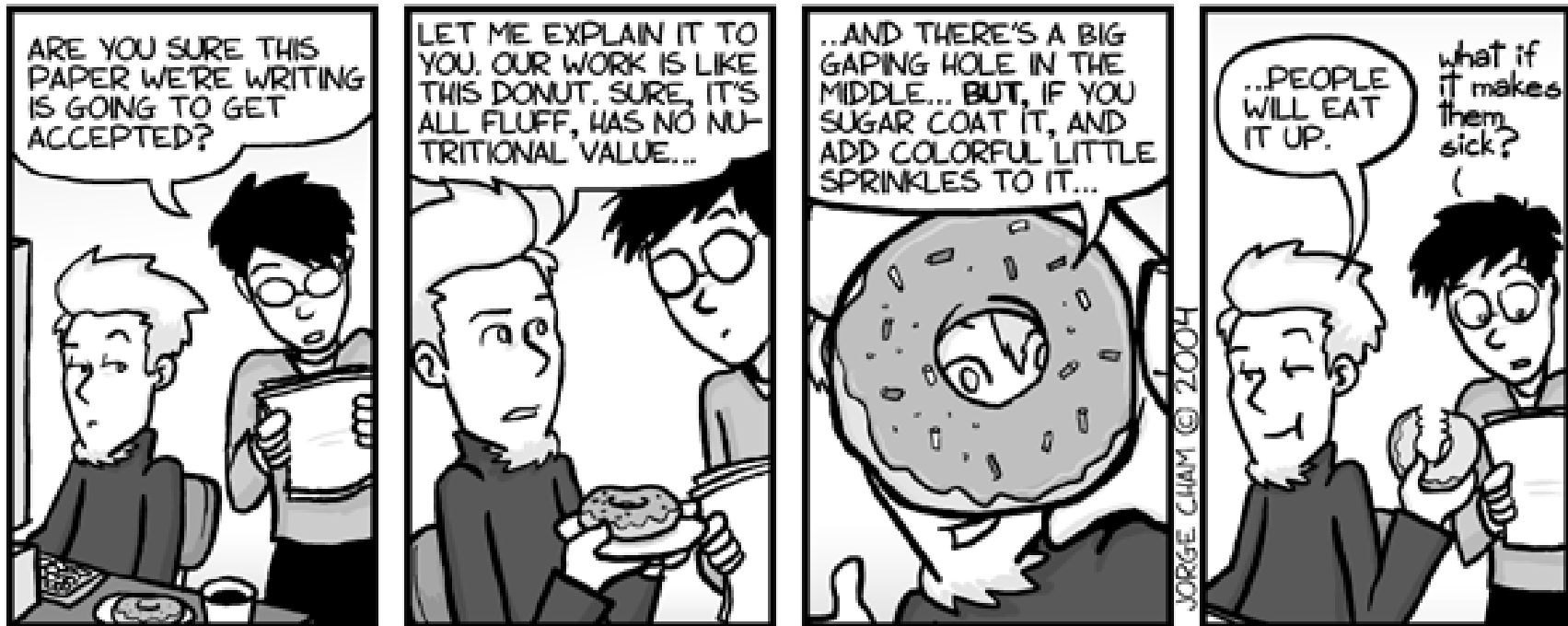
=

"WE HAVE NO CLUE EITHER."

JORGE CHAM © 2004

www.phdcomics.com

In case you did not receive the memo...



www.phdcomics.com

7 Talk Commandments for a Bad Career

1. Always make it sound complicated
2. Never illustrate
3. Never be brief
4. Never print large
5. Never use color
6. Never skip slides in a long talk
7. Always include all your equations and figures from the paper
8. Never, ever, practice

Following all the commandments

- 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

1. Never illustrate
2. Never be brief
3. Never print large
4. Never use color
5. Never try to attract attention
6. Never prepare a short oral overview
7. Never 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.



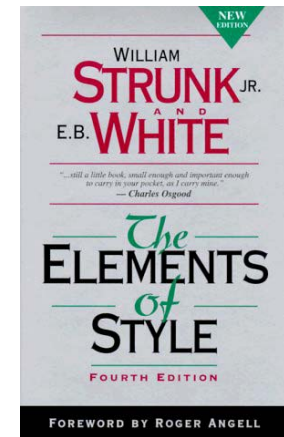
**Part V:
(last, I promise)**

**Alternatives to Bad
Papers/Presentations/Posters**



Alternatives to Bad Papers (1)

- Do the opposite of Bad Paper Commandments
 - Define terms, distinguish "will do" vs "have done", mention drawbacks, real performance, reference other papers.
- Find related work through on-line library catalogs
 - Most papers available on-line
 - ACM and IEEE
 - <http://library.stanford.edu>
- Read the "Elements of Style"
 - Best small book on writing
 - Read it often (every 2 years?)
 - Longer alternative "Bugs in Writing" by Lyn Durpe



Alternatives to Bad Papers (2)

- Follow these steps:
 1. 1-page paper outline, with tentative page budget/section
 2. Paragraph map
 - » 1 topic phrase/sentence per paragraph, hand-drawn 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 advisor, friends, and critics on draft; go to 3.

Alternatives to Bad Talks

- Do the opposite of Bad Talk Commandments
- Allocate 2 minutes per slide, 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
- Bill Tetzlaff, IBM:
 - "Giving a first class 'job talk' is the single most important part of an interview trip. Having someone know that you can give an excellent talk before hand greatly increases the chances of an invitation. That means great conference talks."

Alternatives to Bad Posters (from Randy Katz)

- Do opposite of Bad Poster commandments
 - Poster tries to catch the eye of person walking by
- 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?
- 9 page poster might look like

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

ROC: Recovery-Oriented Computing

Aaron Brown and David Patterson

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

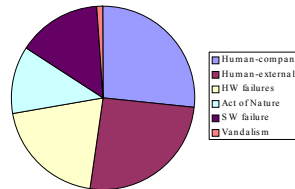
For more info: <http://roc.eecs.berkeley.edu>

AME is the 21st Century Challenge

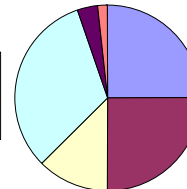
- **Availability**
 - systems should continue to meet quality of service goals despite hardware and software failures
- **Maintainability**
 - systems should require only minimal ongoing human administration, regardless of scale or complexity: Today, cost of maintenance = 10X cost of purchase
- **Evolutionary 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

Number of Outages



Minutes of Failure



- **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 = $\frac{MTTF}{(MTTF + MTTR)}$
 - Since $MTTF \gg MTTR$, 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.

Links to More Advice on Papers/Presentations/Posters

- Patterson's tips on writing
 - <http://www.cs.berkeley.edu/~pattrsn/talks/writingtips.html>
- Armando's paper writing & presentation page
 - http://swig.stanford.edu/~fox/paper_writing.html
- The "How to Give a Talk" Talk
 - <http://www.eecs.harvard.edu/~margo/slides/>
- "The Task of the Referee", A.J Smith
 - Advice on How to Review Papers
 - http://www.cs.wisc.edu/~markhill/the_task_of_the_referee.pdf
- Jason Hong's very extensive list of links
 - Links on research, writing, presentation, thesis
 - <http://www.cs.berkeley.edu/~jasonh/advice.html>

Overall Summary: Enjoy Graduate School

- **Show Initiative!**
 - Don't wait for advisor (or senior students) to show you what to do
 - Selecting your PhD topic is as important as your results (some people say it is even more important)
- **Ask questions!**
 - Lots of smart people in grad school (and even on the faculty), but don't be intimidated.
 - Either they know and you will learn, or they don't know and you will all learn by trying to determine the answer
 - It's OK to ask a "stupid question" every now and then
- **When to graduate**

