How to Communicate Poorly:
giving bad talks, show bad posters,
writing bad papers

Professor David A. Patterson

December 2004

www.cs.berkeley.edu/~pattrsn/talks/nontech.html
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

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.
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.
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 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)

- **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**

<table>
<thead>
<tr>
<th>Problem Statement</th>
<th>State-of-the-Art</th>
<th>Key Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accomplish # 1</td>
<td>Title and Visual logo</td>
<td>Accomplish # 2</td>
</tr>
<tr>
<td>Accomplish # 3</td>
<td>Plan for Success</td>
<td>Summary &amp; Conclusion</td>
</tr>
</tbody>
</table>
**ROC: Recovery-Oriented Computing**

**Aaron Brown and David Patterson**

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

---

**AME is the 21st Century Challenge**
- **Availability**
  - System should continue to meet quality of service goals despite hardware and software failures
- **Maintainability**
  - System 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/updated/expanded
- **Performance was the 20th Century Challenge**
  - 100X Speedup suggests problems are elsewhere

---

**People are the biggest challenge**

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

---

**ROC Principles:**

1. **Isolation and redundancy**
   - System is partitionable
   - Isolate faults
   - To enable online repair/recovery
   - To enable online HW growth/5W upgrade
   - To enable generator training/expand experience on portions of real system
   - Techniques: Geographically replicated sites, Stand-alone cluster, Separate address spaces inside CPU
   - System is redundant
   - Sufficient HW redundancy/data replication as part of system down but satisfactory service still available
   - Enough to survive 2h failure or more during recovery
   - Techniques: RAID-6: N-copies of data

2. **Online verification**
   - System enables input insertion, output check of all modules (including fault insertion)
   - To check module operation to find failure faster
   - To test correctness of recovery mechanisms
   - INSERT faults and known incorrect inputs
   - Also enable availability benchmarks
   - To test if proposed solution fixed the problem
   - Discover whether need to try another solution
   - Discover if warning systems are broken
   - To expose and remove latent errors from each system
   - To train senior experience of operators
   - Techniques: Global monitors, Topology discovery, Program checking (SW/ECC)

3. **Undo Support**
   - ROC system should offer Undo
   - To recover from operator errors
   - To undo all changes in productivity apps
   - Should have `undo for maintenance`
   - To recover from inevitable 5W errors
   - To 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

---

**ROC Principles:**

4. **Diagnosis Support**
   - System assists in diagnosing problems
   - Root-cause analysis to suggest possible failure points
   - Track resource dependencies of all requests
   - Create symptom correlation with component dependency model to isolate culprit component
   - 'Health' reporting to detect failed/failing components
   - Fault information, self-test results propagated upward
   - Unified status console to highlight improper behavior, predict failure, and suggest corrective action
   - Techniques: Start data blocks with module IDs; Log faults, errors, failures and recovery methods

---

**Lessons Learned from Other Fields**

- 1800s: 25% railroad bridges failed!

  **Henry Petroski**

  Engineering and the Human Dilemma

  - Techniques invented:
    - 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 is 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

---

For more info: http://roc poderá ser
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
  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
Administrivia

• Announcement of Final Project Presentation times
  – Tuesday, 10:30 AM to 1 PM, Soda 611 (west alcove),
  – please email your times your group CANNOT make to TA

• Processor racing is at 7:30 Wed Dec 7 in the lab,
  – with LaVals afterwards at 9PM

• Reflections on Xilinx visit?