



# Development Time Working Group

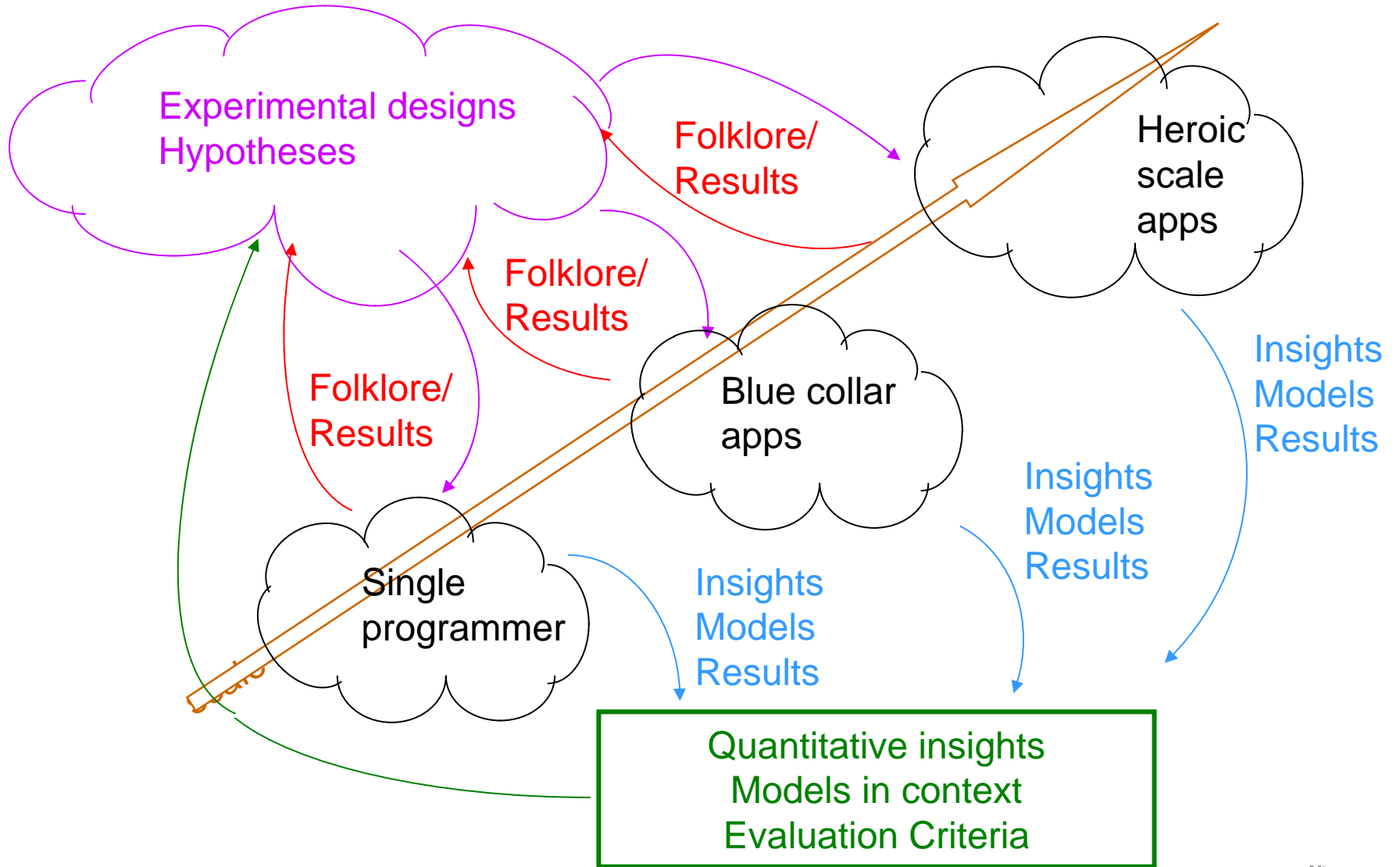
**Jeff Hollingsworth**

**HPCS Workshop  
November 2005**





# From Hypotheses to Insight





# Development Group Goals



- Provide empirical evidence for assumptions made by HPC community about development time issues
- Evolve a series of experiments and case studies
- Develop criteria for HPC systems productivity evaluation
- Better understand software development for HPC
- Obtain a more complete view of the total time to solution
- Provide better guidance for planning and decision-making in HPC code development



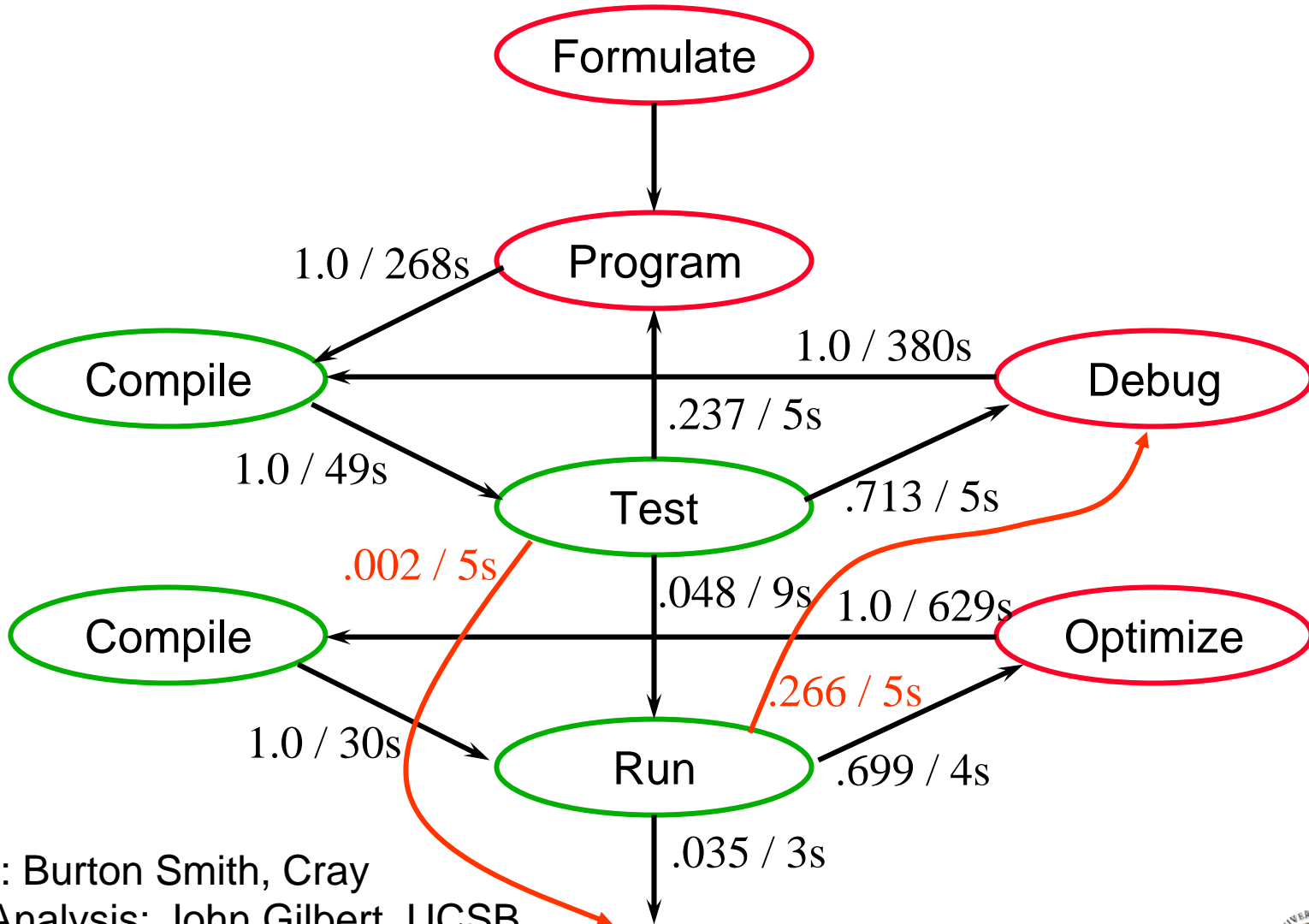


# Workflow of Programming Activities



		Effort Log & Measured Data			
		P1	P2	P3	P4
<b>Physical Activities</b>  <b>Logical Activities</b>		Formulate	Editing	Compiling	Running
1	Planning, Designing	planning, designing			
2	Understanding the Environment	Experimenting with environment			
3	Serial Coding	Serial coding			
4	Parallel Coding	Parallelizing the code			
5	Testing/Debugging	Debugging			
		Testing			
6	Tuning/Optimizing	Tuning			





Model: Burton Smith, Cray  
 Data Analysis: John Gilbert, UCSB



# Classroom Studies: Overview



- Approach:
  - Gather data from students taking an graduate HPC class
    - Effort Logs
    - Program snapshots at every compile
    - Sometimes every keystroke
  - Learn where programmers spend their time to make things better
  - Conducted studies in 12 classes, 2 ongoing
- Programming models investigated:
  - MPI, OpenMP, Co-Array Fortran, UPC, Matlab\*P, XMT-C.
- Analysis to date has focused on
  - input: programming model, problem
  - output: programmer effort, program performance
- Upcoming analysis will focus on
  - Activities/workflow - identifying development activities from data
  - Defects - analyzing recurring defect patterns





# Current classroom studies



- Alan Sussman @ UMD
  - 1 Assignment: sharks & fishes
    - Half students do MPI, half do OpenMP, then switch
  - Class projects
    - Perform small case studies to understand scale-up, debug methodology before larger case studies
- Aiichiro Nakano @ USC
  - Improve MPI molecular dynamics program by adding OpenMP
- Analysis will focus on
  - defect-related issues
  - effort and workflows





# Development Time Studies: Sample Results

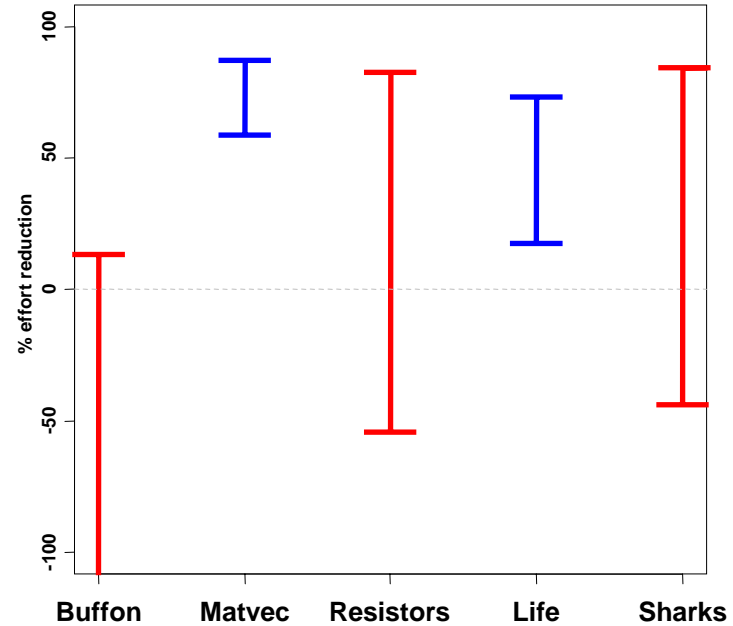


Effect of model & problem on

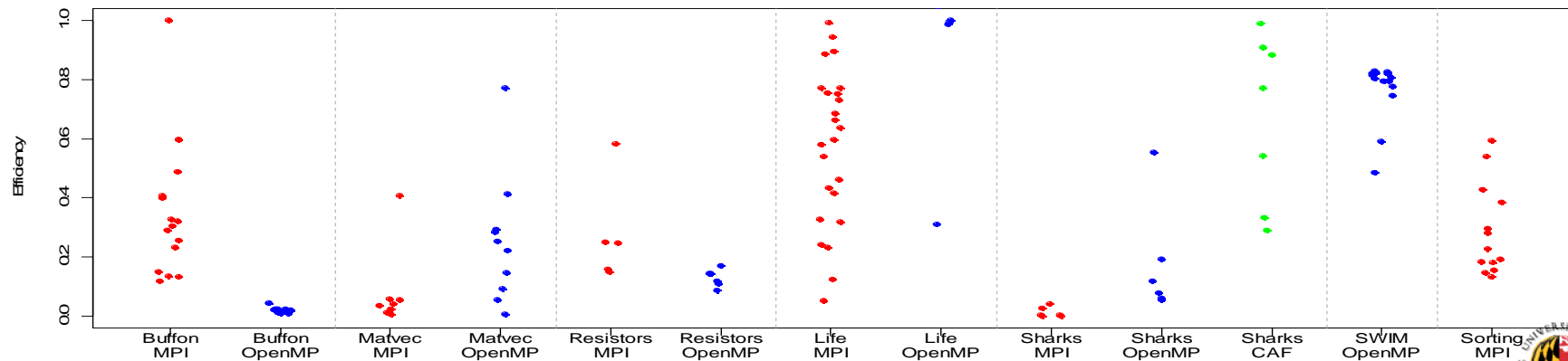
% Effort saved using OpenMP instead of MPI

Effort

Performance



Performance (efficiency) by problem/language





# Development Time Results



## For Classroom Studies:

- 35-75% time savings using OpenMP vs. MPI
  - Appears to scale linearly with problem size
- Experience with a problem improves time to solution, but
  - Programming model has a larger impact
- When performance is the goal:
  - Experts and students spend the same amount of time
  - Experts get orders of magnitude better performance
- Have not seen productivity gain from UPC/CAF vs. MPI
  - May be due to size of classroom studies
- No correlation between effort and performance





# Defects/Quality Analysis



- Contribution to research
  - Classifying defects to characterize recurring problems in HPC development
    - Understanding how novices and experts of HPC development are similar/different with respect to problems encountered
- Contribution to project
  - Minimizing HPC development time by reducing defects and taking efficient problem-solving approach
    - Identifying techniques and tools for predicting, detecting, and preventing potential HPC defects
- Current status
  - Performed preliminary source code analysis on novice data to confirm the existence of distinctive patterns in problem-solving and associated defects
  - Preliminary analysis on experts data is in progress





# Research Questions



- What do real HPC defects look like?
  - Can we identify recurring defects? Are there patterns?
  - Can we identify defect patterns relationships:
    - To language and problem domain?
  - Can we define HPC defect classifications?
    - Completeness, orthogonally, consistency, usefulness
- What are the relationship between defects and workflow?
  - When are defects created, found and fixed during HPC development?
  - Can we characterize defects with respect to phase and activity they occur?
  - Which defects take more efforts to deal with?
- What can prevent HPC defects?
  - Can the prior knowledge on recurring HPC defects prevent developer from making them?
  - What kind of tools can detect a certain class of HPC defects?



- “mandatory” MPI operations

```
#include <mpi.h>

MPI_Init(...);
MPI_Comm_size(...);
MPI_Comm_rank(...);
read initial cell states from input file

for (# of iterations) {
  for (each cell) {
    if (# of neighbors is 2) do nothing
    if (# of neighbors is 3) occupy
    otherwise empty
  }
}

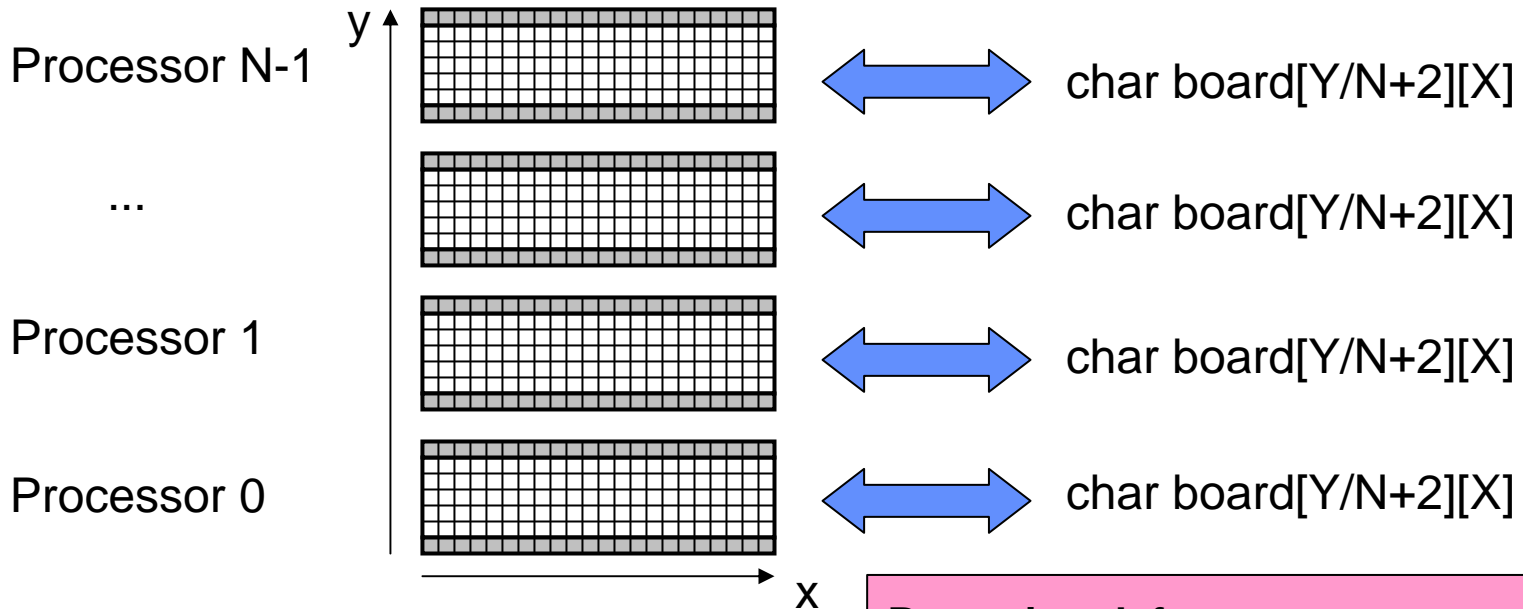
write final cell states to output file
MPI_Finalize();
```

## Recurring defects:

- Improper API parameters
- Missing MPI\_Finalize()
- Unchecked return values



- Implementation of main iteration loop
  - Additional buffers for storing send/recv data

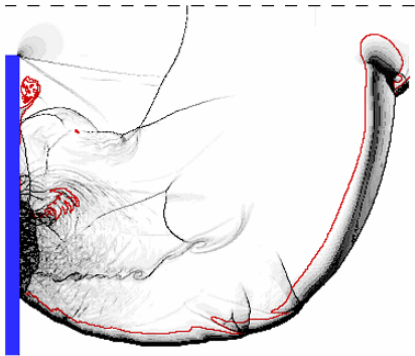


```

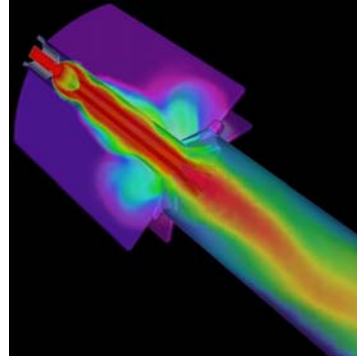
for (# of iterations) {
    update cells
    MPI_Send (send to processor (i+1));
    MPI_Recv (recv to processor (i-1));
    MPI_Send (send to processor (i-1));
    MPI_Recv (recv to processor (i+1));
}
    
```

**Recurring defects:**

- Incorrect conversion logic between board coordinates and array index,
- Improper use/omission of “Barriers”



Caltech



UIUC



Utah

- Research contribution: *Understand similarities & differences of mature, in-the-large HPC projects, contrast with “toy” problems in class experiments*
- Project contribution: *Study productivity issues in projects which are future users of HPCS systems*
- Status: *Conducted interviews with project leads, looking at defect data*



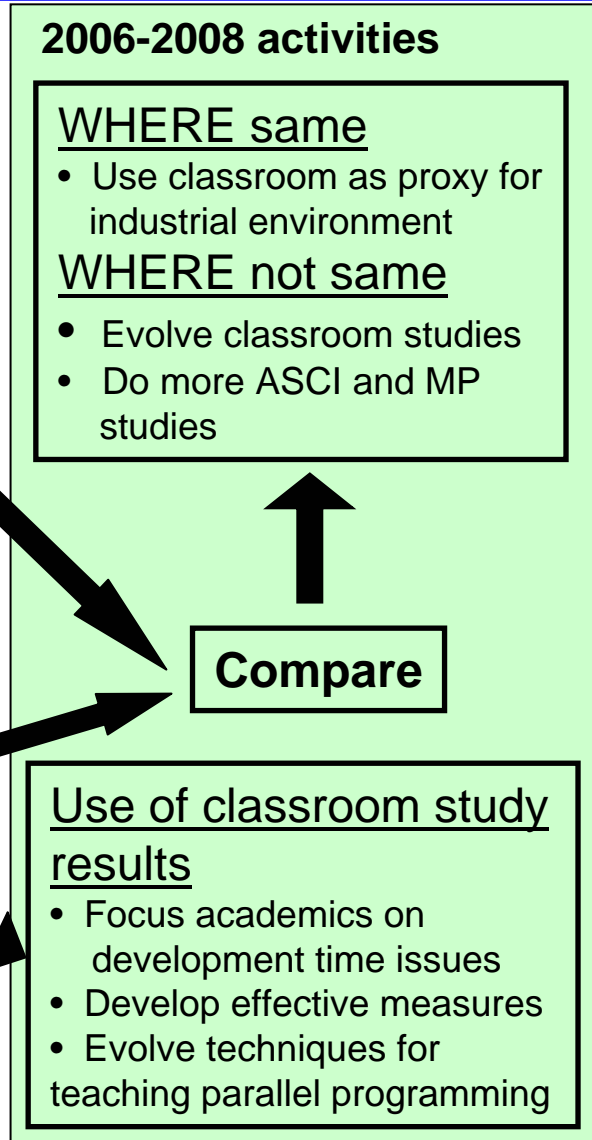
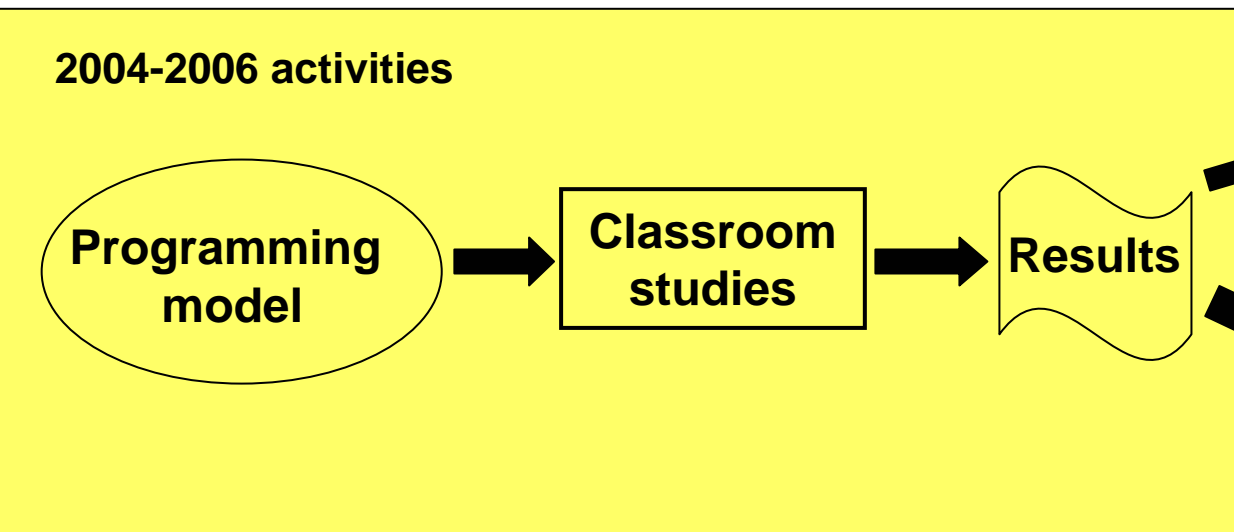
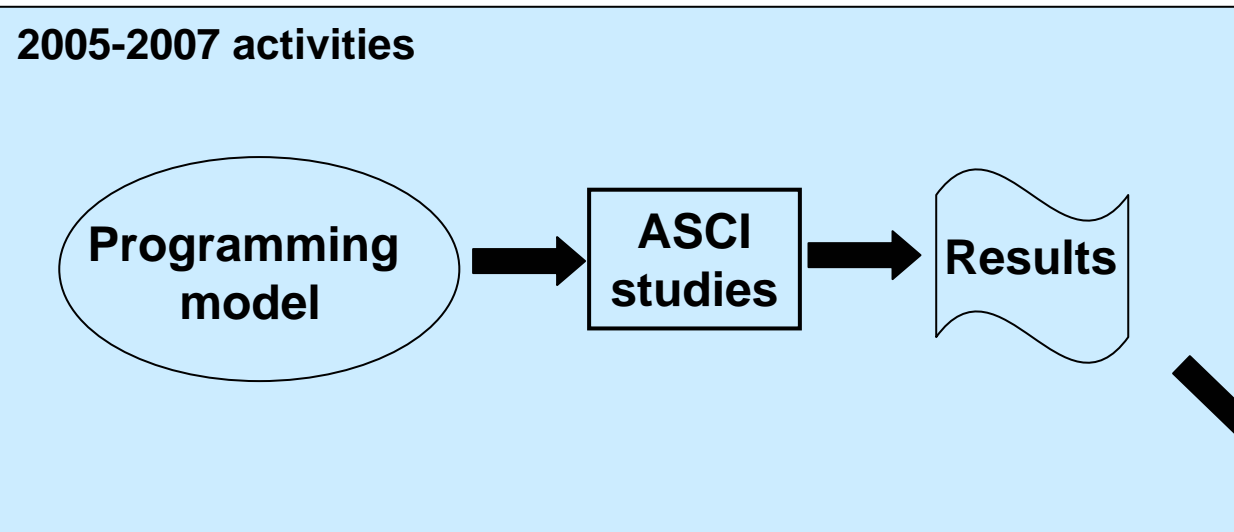
# Case Study: Observed Trends



- Code:
  - 80-530 KLOC; 2-10 external libraries
  - All contain 4-5 coupled components
  - All Use MPI
  - All codes close to linear ISO-scaling to 1,000 nodes
- Development Teams:
  - 10-25 core team (faculty, postdocs, staff, grad students)
    - 5-25 years programming experience (3-15 on parallel systems)
  - Designated “tuner”: mainly system independent tuning
    - Don’t want to make changes for any one platform – it will vanish soon
- Process:
  - All use revision control system (to varying degrees)
  - No formal defect tracking process



- Unique Aspects of Teams (relative to industry teams)
  - Developers are also major users
  - Continual evolution of algorithms
    - based on program output
  - Relatively low QA effort: 10% of project resources
- Teams' Self Observations
  - Would prefer to write equations, not code, not MPI
  - Debuggers not useful for algorithmic debugging
    - Don't work well in a remote environment in general





# Publications (1)



## 2004

- **Challenges in Measuring HPCS Learner Productivity in an Age of Ubiquitous Computing: The HPCS Program**  
In Proceedings of First International Workshop on Software Engineering for High Performance Computing System Applications, Edinburgh, Scotland, May 2004, pp. 27-31.
- **Studying Code Development for High Performance Computing: The HPCS Program**  
In Proceedings of First International Workshop on Software Engineering for High Performance Computing System Applications, Edinburgh, Scotland, May 2004, pp. 32-36.

## 2005

- **Generating Testable Hypotheses from Tacit Knowledge for High Productivity Computing**  
In Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications, St. Louis, MO, May 2005.
- **Application of a Development Time Productivity Metric to Parallel Software Development**  
In Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications, St. Louis, MO, May 2005.
- **A Metric Space for Productivity Measurement in Software Development**  
In Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications, St. Louis, MO, May 2005.





# Publications (2)



## 2005

- **Combining self-reported and automatic Data to improve programming effort measurement**  
Presented at Foundations of Software Engineering (FSE05) Lisbon, Portugal, September 2005
- **Measuring Productivity on High Performance Computers**  
Presented at International Symposium on Software Metrics, Como, Italy, September 2005.
- **Empirical study design in the area of High-Performance Computing (HPC)**  
To be presented at the International Symposium of Empirical Software Engineering (ISESE05), Noosa Heads, Australia, November 2005.
- **HPC programmer productivity: a case study of novice HPC programmers**  
To be presented at Super Computing (SC2005) Seattle, WA, November 2005

## 2006

- **Using folklore to build models and hypotheses in high performance computing**  
Submitted to ICSE 2006.
- **An empirical study to compare the productivity of two parallel programming models**  
Submitted to ICSE 2006

**Two students working on HPCS as their Ph.D. topic:  
Lorin Hochstein, Taiga Nakamura**

