HPCS Application Analysis and Assessment

Dr. Jeremy Kepner / Lincoln
Dr. David Koester / MITRE

This work is sponsored by the Department of Defense under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.
Outline

- Motivation
- Productivity Framework

- Introduction
- Workflows
- Metrics
- Models & Benchmarks
- Schedule and Summary
Create a new generation of economically viable computing systems and a procurement methodology for the security/industrial community (2007 – 2010)

Full Scale Development

Advanced Design & Prototypes

Concept Study

Phase 1
$20M (2002)

Phase 2
$180M (2003-2005)

Phase 3
(2006-2010)

2 Vendors

Validated Procurement Evaluation Methodology

Test Evaluation Framework

New Evaluation Framework

MITRE

MIT Lincoln Laboratory

ISI
Motivation: Metrics Drive Designs
“You get what you measure”

**Execution Time (Example)**
- Current metrics favor caches and pipelines
  - Systems ill-suited to applications with
    - Low spatial locality
    - Low temporal locality

**Development Time (Example)**
- No metrics widely used
  - Least common denominator standards
  - Difficult to use
  - Difficult to optimize

- **HPCS needs a validated assessment methodology that values the “right” vendor innovations**
- **Allow tradeoffs between Execution and Development Time**
Phase 1: Productivity Framework

Productivity (Ratio of Utility/Cost)

- Work Flows
- Execution Time (cost)
- Development Time (cost)

Productivity Metrics

- Activity & Purpose Benchmarks
- System Parameters (Examples)
  - BW bytes/flop (Balance)
  - Memory latency
  - Memory size
  - Processor flop/cycle
  - Processor integer op/cycle
  - Bisection BW
  - Size (ft³)
  - Power/rack
  - Facility operation
  - Code size
  - Restart time (Reliability) Code
  - Optimization time

Actual System or Model

Common Modeling Interface

Activity & Purpose Benchmarks

Slide-5
HPCS Productivity

MITRE
MIT Lincoln Laboratory
ISI
Phase 2: Implementation

**Productivity**
(Ratio of Utility/Cost)

**Productivity Metrics**

**Development Time (cost)**

**Execution Time (cost)**

**Work Flows**

**Activity & Purpose**

**Benchmarks**

**System Parameters (Examples)**
- BW bytes/flop (Balance)
- Memory latency
- Memory size
- Processor flop/cycle
- Processor integer op/cycle
- Bisection BW
- Size (ft³)
- Power/rack
- Facility operation
- Code size
- Restart time (Reliability) Code
- Optimization time

**Common Modeling Interface**

**Metrics Analysis of Current and New Codes**
(Lincoln, UMD & Mission Partners)

**Metrics Analysis of University Experiments**
(MIT, UCSB, UCSD, UMD, USC)

**Performance Analysis**
(ISI, LLNL & UCSD)

**Actual System or Model**

**Exe Interface**

**Dev Interface**

(Lincoln, OSU, CodeSourcery)

(Mitre, ISI, LBL, Lincoln, HPCMO, LANL & Mission Partners)
Outline

• Introduction
• Workflows
• Metrics
• Models & Benchmarks
• Schedule and Summary

• Lone Researcher
• Enterprise
• Production
HPCS Mission Work Flows

Overall Cycle

Development Cycle

HPCS Productivity

Factors: Performance, Programmability, Portability, and Robustness are very closely coupled with each work flow
• Missions (development): Cryptanalysis, Signal Processing, Weather, Electromagnetics

• Process Overview
  – Goal: solve a compute intensive domain problem: crack a code, incorporate new physics, refine a simulation, detect a target
  – Starting point: inherited software framework (~3,000 lines)
  – Modify framework to incorporate new data (~10% of code base)
  – Make algorithmic changes (~10% of code base); Test on data; Iterate
  – Progressively increase problem size until success
  – Deliver: code, test data, algorithm specification

• Environment overview
  – Duration: months Team size: 1
  – Machines: workstations (some clusters), HPC decreasing
  – Languages: FORTRAN, C → Matlab, Python
  – Libraries: math (external) and domain (internal)

• Software productivity challenges
  – Focus on rapid iteration cycle
  – Frameworks/libraries often serial
Domain Researcher (special case)

- **Scientific Research:** DoD HPCMP Challenge Problems, NNSA/ASCI Milestone Simulations

- **Process Overview**
  - Goal: Use HPC to perform Domain Research
  - Starting point: Running code, possibly from an Independent Software Vendor (ISV)
  - NO modifications to codes
  - Repeatedly run the application with user defined optimization

- **Environment overview**
  - Duration: months, Team size: 1-5
  - Machines: workstations (some clusters), HPC
  - Languages: FORTRAN, C
  - Libraries: math (external) and domain (internal)

- **Software productivity challenges — None!**

- **Productivity challenges**
  - Robustness (reliability)
  - Performance
  - Resource center operability
Enterprise Design

• Missions (development): Weapons Simulation, Image Processing

• Process Overview
  – Goal: develop or enhance a system for solving a compute intensive domain problem: incorporate new physics, process a new surveillance sensor
  – Starting point: software framework (~100,000 lines) or module (~10,000 lines)
  – Define sub-scale problem for initial testing and development
  – Make algorithmic changes (~10% of code base); Test on data; Iterate
  – Progressively increase problem size until success
  – Deliver: code, test data, algorithm specification, iterate with user

• Environment overview
  – Duration: ~1 year Team size: 2-20
  – Machines: workstations, clusters, hpc
  – Languages: FORTRAN, C, → C++, Matlab, Python, IDL
  – Libraries: open math and communication libraries

• Software productivity challenges
  – Legacy portability essential
    Avoid machine specific optimizations (SIMD, DMA, …)
  – Later must convert high level language code
Production

- Missions (production): Cryptanalysis, Sensor Processing, Weather

- Process Overview
  - Goal: develop a system for fielded deployment on an HPC system
  - Starting point: algorithm specification, test code, test data, development software framework
  - Rewrite test code into development framework; Test on data; Iterate
  - Port to HPC; Scale; Optimize (incorporate machine specific features)
  - Progressively increase problem size until success
  - Deliver: system

- Environment overview
  - Duration: ~1 year  Team size: 2-20
  - Machines: workstations and HPC target
  - Languages: FORTRAN, C, → C++

- Software productivity challenges
  - Conversion of higher level languages
  - Parallelization of serial library functions
  - Parallelization of algorithm
  - Sizing of HPC target machine
Production Workflow

- Many technologies targeting specific pieces of workflow
- Need to quantify workflows (stages and % time spent)
- Need to measure technology impact on stages

<table>
<thead>
<tr>
<th>Algorithm Development</th>
<th>Spec</th>
<th>Design, Code, Test</th>
<th>Port, Scale, Optimize</th>
<th>Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supercomputer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operating Systems
- Linux
- RT Linux

Compilers
- Matlab
- Java
- C++
- OpenMP
- F90
- UPC Coarray
- ATLAS, BLAS,
- FFTW, PETE, PAPI

Libraries
- CORBA
- VSIPL
- VSIPL++
- MPI
- DRI
- Globus
- TotalView

Tools
- UML
- ATLAS, BLAS,
- FFTW, PETE, PAPI

Problem Solving Environments
- Matlab
- Java
- C++
- OpenMP
- F90
- UPC Coarray
- ATLAS, BLAS,
- FFTW, PETE, PAPI

Mainstream Software
- CORBA
- VSIPL
- VSIPL++
- MPI
- DRI
- Globus
- TotalView

HPC Software
- CCA
- ESMF
- POOMA
- PVL

MITRE
MIT Lincoln Laboratory
ISI
Example: Coding vs. Testing

Workflow Breakdown (NASA SEL)

<table>
<thead>
<tr>
<th></th>
<th>Analysis and Design</th>
<th>Coding and Auditing</th>
<th>Checkout and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sage</td>
<td>39%</td>
<td>14%</td>
<td>47%</td>
</tr>
<tr>
<td>NTDS</td>
<td>30</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>Gemini</td>
<td>36</td>
<td>17</td>
<td>47</td>
</tr>
<tr>
<td>Saturn V</td>
<td>32</td>
<td>24</td>
<td>44</td>
</tr>
<tr>
<td>OS/360</td>
<td>33</td>
<td>17</td>
<td>50</td>
</tr>
<tr>
<td>TRW Survey</td>
<td>46</td>
<td>20</td>
<td>34</td>
</tr>
</tbody>
</table>

Testing Techniques (UMD)

**Code Reading**
- Reading by Stepwise Abstraction

**Functional Testing**
- Boundary Value Equivalence Partition Testing

**Structural Testing**
- Achieving 100% statement coverage

What is HPC testing process?

<table>
<thead>
<tr>
<th>Environment</th>
<th>Small (Workstation)</th>
<th>Medium (Cluster)</th>
<th>Full (HPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype (Matlab)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial (C/Fortran)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel (OpenMP)</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

New Result? New Bug?
Outline

- Introduction
- Workflows
- **Metrics**
  - Existing Metrics
  - Dev. Time Experiments
  - Novel Metrics
- Models & Benchmarks
- Schedule and Summary
Analysis of existing codes used to test metrics and identify important trends in productivity and performance.
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Serial Fortran</td>
</tr>
<tr>
<td>BT</td>
<td>✔️</td>
</tr>
<tr>
<td>CG</td>
<td>✔️</td>
</tr>
<tr>
<td>EP</td>
<td>✔️</td>
</tr>
<tr>
<td>FT</td>
<td>✔️</td>
</tr>
<tr>
<td>IS</td>
<td>✔️</td>
</tr>
<tr>
<td>LU</td>
<td>✔️</td>
</tr>
<tr>
<td>MG</td>
<td>✔️</td>
</tr>
<tr>
<td>SP</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Source Lines of Code (SLOC) for the NAS Parallel Benchmarks (NPB)

Serial Implementation (Fortran / C)

Benchmark

SLOC

BT CG EP FT IS LU MG SP

Source Lines of Code (SLOC) for the NAS Parallel Benchmarks (NPB)
Normalized SLOC for All Implementations of the NPB
NAS FT Performance vs. SLOCs

- **Fortran / MPI, 16 Processors**
- **Java, 16 Processors**
- **Serial Fortran, 1 Processor**

Performance (Mops) vs. Development Effort (SLOC)
Controlled experiments can potentially measure the impact of different technologies and quantify development time and execution time tradeoffs.
Novel Metrics

- HPC Software Development often involves changing code ($\Delta x$) to change performance ($\Delta y$)
  - 1st order size metrics measures scale of change $E(\Delta x)$
  - 2nd order metrics would measure nature of change $E(\Delta x^2)$

- Example: 2 Point Correlation Function
  - Looks at “distance” between code changes
  - Determines if changes are localized (good) or distributed (bad)

- Other Zany Metrics
  - See Cray talk
Outline

- Introduction
- Workflows
- Metrics
  - Models & Benchmarks
  - Prototype Models
  - A&P Benchmarks
- Schedule and Summary
Prototype Productivity Models

### Special Model with Work Estimator (Sterling)

\[ \Psi_w = \frac{SP \times E \times A}{Cf \times \left\{ \sum \left( \rho \cdot n \right) \right\} + \left( c_m + c_o \right) \times T} \]

### Utility (Snir)

\[ P(S, A, U(.)) = \min_{cost} \frac{U(T(S, A, Cost))}{Cost} \]

### Productivity Factor Based (Kepner)

- Productivity \( \approx \frac{\text{useful ops}}{\text{second}} \cdot \frac{\text{GUPS}}{\text{Linpack}} \cdot \frac{\text{efficiency factor}}{\text{mission factor}} \)
- \( \text{productivity factor} \approx \left( \frac{\text{Language Level}}{\text{Parallel Model}} \right) \times \frac{\text{Portability}}{\text{Availability}} \times \frac{\text{Maintenance}}{\text{Factor}} \)

### HPCS has triggered ground breaking activity in understanding HPC productivity

- Community focused on quantifiable productivity (potential for broad impact)
- Numerous proposals provide a strong foundation for Phase 2
**Code Size and Reuse Cost**

<table>
<thead>
<tr>
<th>Lines of code</th>
<th>Function Points</th>
<th>Reuse</th>
<th>Re-engineering</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Size</td>
<td>Code = ( \text{New} + \text{Reused} + \text{Re-engineered} + \text{Maintained} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Measured in lines of code or functions points (converted to lines of code)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lines per function point</th>
<th>C, Fortran</th>
<th>~100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fortran77</td>
<td>~100</td>
</tr>
<tr>
<td></td>
<td>C++</td>
<td>~30</td>
</tr>
<tr>
<td></td>
<td>Java</td>
<td>~30</td>
</tr>
<tr>
<td></td>
<td>Matlab</td>
<td>~10</td>
</tr>
<tr>
<td></td>
<td>Python</td>
<td>~10</td>
</tr>
<tr>
<td></td>
<td>Spreadsheet</td>
<td>~5</td>
</tr>
</tbody>
</table>

**HPC Challenge Areas**

**Function Points**
- High productivity languages not available on HPC

**Reuse**
- Nonlinear reuse effects. Performance requirements dictate “white box” reuse model

- Code size is the most important software productivity parameter
- Non-HPC world reduces code size by
  - Higher level languages
  - Reuse
- HPC performance requirements currently limit the exploitation of these approaches
Activity & Purpose Benchmarks define a set of instructions (i.e., source code) to be executed. Purpose Benchmarks define requirements, inputs, and output. Together, they address the entire development workflow.
### HPCS Phase 1 Example Kernels and Applications

<table>
<thead>
<tr>
<th>Mission Area</th>
<th>Kernels</th>
<th>Application</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stockpile Stewardship</td>
<td>Random Memory Access</td>
<td>UMT2000</td>
<td>ASCI Purple Benchmarks</td>
</tr>
<tr>
<td></td>
<td>Unstructured Grids</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eulerian Hydrocode</td>
<td>SAGE3D</td>
<td>ASCI Purple Benchmarks</td>
</tr>
<tr>
<td></td>
<td>Adaptive Mesh</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unstructured Finite Element Model</td>
<td>ALEGRA</td>
<td>Sandia National Labs</td>
</tr>
<tr>
<td></td>
<td>Adaptive Mesh Refinement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operational Weather and Ocean</td>
<td>Finite Difference Model</td>
<td>NLOM</td>
<td>DoD HPCMP TI-03</td>
</tr>
<tr>
<td>Forecasting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Army Future Combat Weapons Systems</td>
<td>Finite Difference Model</td>
<td>CTH</td>
<td>DoD HPCMP TI-03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crashworthiness Simulations</td>
<td>Multiphysics Nonlinear Finite Element</td>
<td>LS-DYNA</td>
<td>Available to Vendors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Kernels</td>
<td>Lower / Upper Triangular Matrix Decomposition</td>
<td>LINPACK</td>
<td>Available on Web</td>
</tr>
<tr>
<td></td>
<td>Conjugate Gradient Solver</td>
<td>DoD HPCMP TI-03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>QR Decomposition</td>
<td></td>
<td>Paper &amp; Pencil for Kernels</td>
</tr>
<tr>
<td></td>
<td>Biological Pathway Analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Bio-Application

<table>
<thead>
<tr>
<th>Kernels</th>
<th>Application</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantum and Molecular Mechanics</td>
<td>Macromolecular Dynamics</td>
<td>CHARMM <a href="http://yuri.harvard.edu/">http://yuri.harvard.edu/</a></td>
</tr>
<tr>
<td></td>
<td>Energy Minimization</td>
<td>MonteCarlo Simulation</td>
</tr>
<tr>
<td>Whole Genome Analysis</td>
<td>Sequence Comparison</td>
<td>Needleman-Wunsch <a href="http://www.med.nyu.edu/">http://www.med.nyu.edu/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rcr/rcr/course/sim-sw.html</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FASTA <a href="http://www.ebi.ac.uk/fasta33/">http://www.ebi.ac.uk/fasta33/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HMMR <a href="http://hmmer.wustl.edu/">http://hmmer.wustl.edu/</a></td>
</tr>
</tbody>
</table>

**Set of scope benchmarks representing Mission Partner and emerging Bio-Science high-end computing requirements**
Outline

• Introduction
• Workflows
• Metrics
• Models & Benchmarks
• Schedule and Summary
Phase II Productivity Forum
Tasks and Schedule

<table>
<thead>
<tr>
<th>Task (Communities)</th>
<th>FY03</th>
<th>FY04</th>
<th>FY05</th>
<th>FY06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Models</td>
<td>Q3-Q4</td>
<td>Q1-Q2</td>
<td>Q3-Q4</td>
<td>Q1-Q2</td>
</tr>
<tr>
<td>(Lincoln/HPCMO/LANL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev Time Experiments</td>
<td>(UMD/)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev &amp; Exe Interfaces</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(HPC SW/FFRDC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A&amp;P Benchmarks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Missions/FFRDC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified Model Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(HPC Modelers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Experiments</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Modelers/Vendors)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models &amp; Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Modelers/Vendors)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPC Productivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competitiveness Council</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Workflow Models
  - Competing Development Time Models
  - Analyze Existing, Design Exp, & Pilot Studies
  - Controlled Baseline Experiments
  - Mission Specific & New Technology Demonstrations

- Dev Time Experiments
  - Reqs & Spec (~6)
  - Revise & Exe Spec (~2)

- Dev & Exe Interfaces
  - Prototype Interfaces (v0.1)
    - (version 0.5)
    - (version 1.0)

- Machine Experiments
  - Existing HPC Systems
  - Next Generation HPC Systems
  - HPCS Designs

- Models & Metrics
  - Productivity Workshops
  - Productivity Evaluations
  - Roll Out Productivity Metrics

- HPC Productivity Competitiveness Council
  - Validated Dev Time Assessment Methodology
  - Data
  - Workflows
  - Validated Exe Time Assessment Methodology
  - Broad Commercial Acceptance

- Intelligence
- Weapons Design
- Surveillance
- Environment
- Bioinformatics

- Workflows
- Data
Summary

• Goal is to develop an acquisition quality framework for HPC systems that includes
  – Development time
  – Execution time

• Have assembled a team that will develop models, analyze existing HPC codes, develop tools and conduct HPC development time and execution time experiments

• Measures of success
  – Acceptance by users, vendors and acquisition community
  – Quantitatively explain HPC rules of thumb:
    "OpenMP is easier than MPI, but doesn’t scale a high"
    "UPC/CAF is easier than OpenMP"
    "Matlab is easier the Fortran, but isn’t as fast"
  – Predict impact of new technologies
Backup Slides
HPCS Phase II Teams

Industry:

- IBM
- Sun Microsystems
- CRAY

Goal:
- Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2007 – 2010)

Productivity Team (Lincoln Lead)

- MIT Lincoln Laboratory
- ISI
- University of Maryland
- UCSD

Goal:
- Develop a procurement quality assessment methodology that will be the basis of 2010+ HPC procurements
HPCS needs to develop a procurement quality assessment methodology that will be the basis of 2010+ HPC procurements.