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

- **Introduction**
  - ***Motivation***
  - ***Productivity Framework***
- **Workflows**
- **Metrics**
- **Models & Benchmarks**
- **Schedule and Summary**

High Productivity Computing Systems
-Program Overview-

- Create a new generation of economically viable computing systems and a procurement methodology for the security/industrial community (2007 – 2010)

Full Scale Development

Petascale Systems

2 Vendors

Validated Procurement Evaluation Methodology

Advanced Design & Prototypes

CRAY IBM Sun microsystems

team

Test Evaluation Framework

Concept Study

CRAY hp IBM sgi Sun microsystems

MITRE team

New Evaluation Framework

Phase 1
$20M (2002)

Phase 2
$180M (2003-2005)

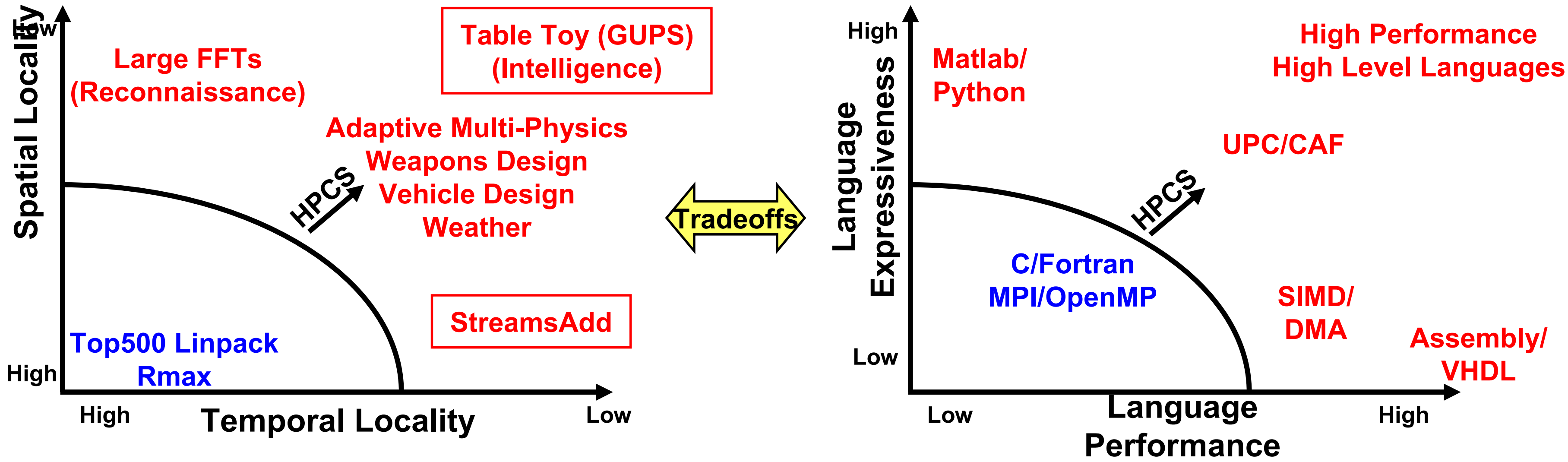
Phase 3
(2006-2010)

MITRE — MIT Lincoln Laboratory — ISI

Motivation: Metrics Drive Designs

"You get what you measure"

Execution Time (Example) | Development Time (Example)



Current metrics favor caches and pipelines

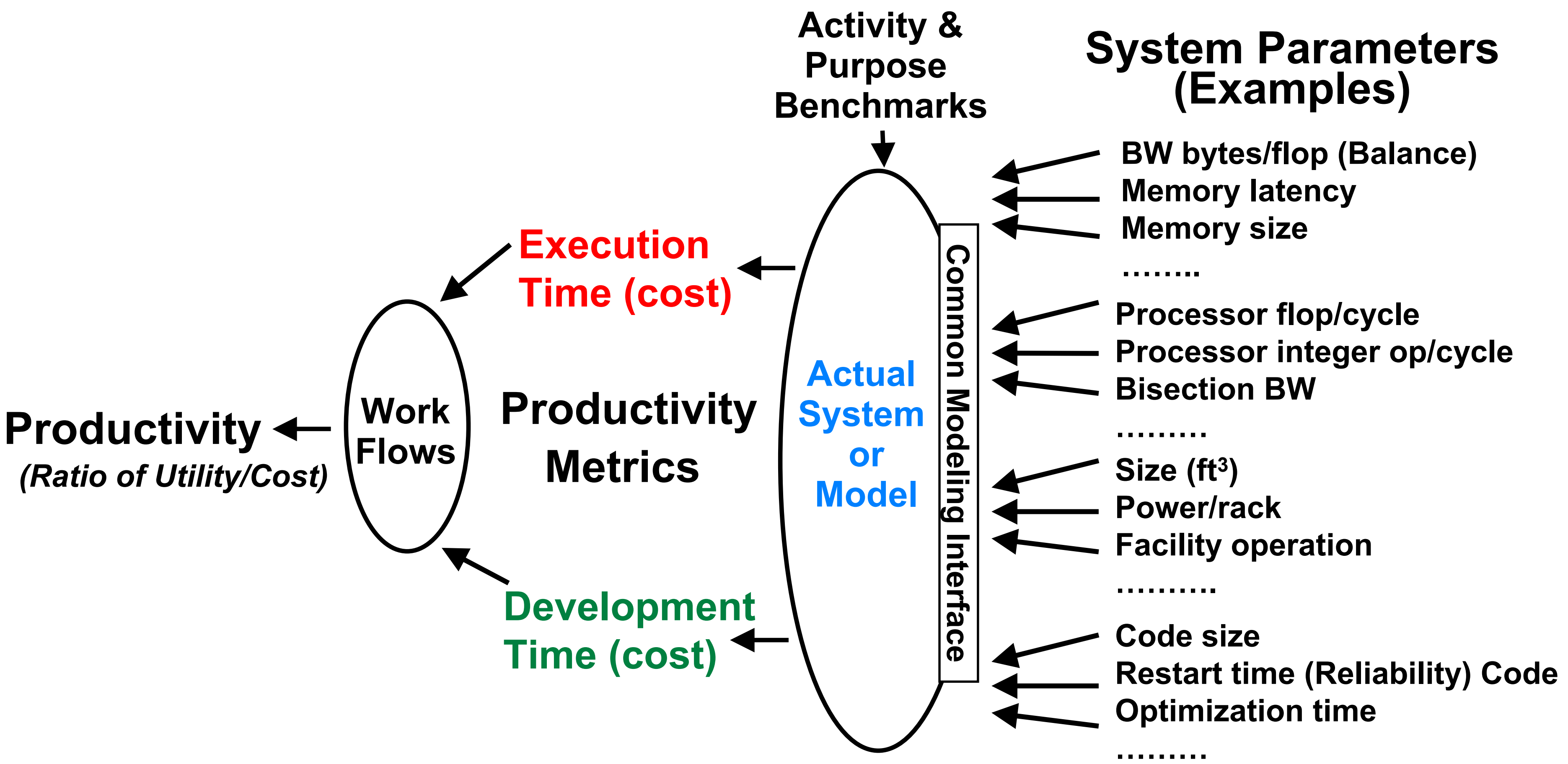
- Systems ill-suited to applications with
- Low spatial locality
- Low temporal locality

No metrics widely used

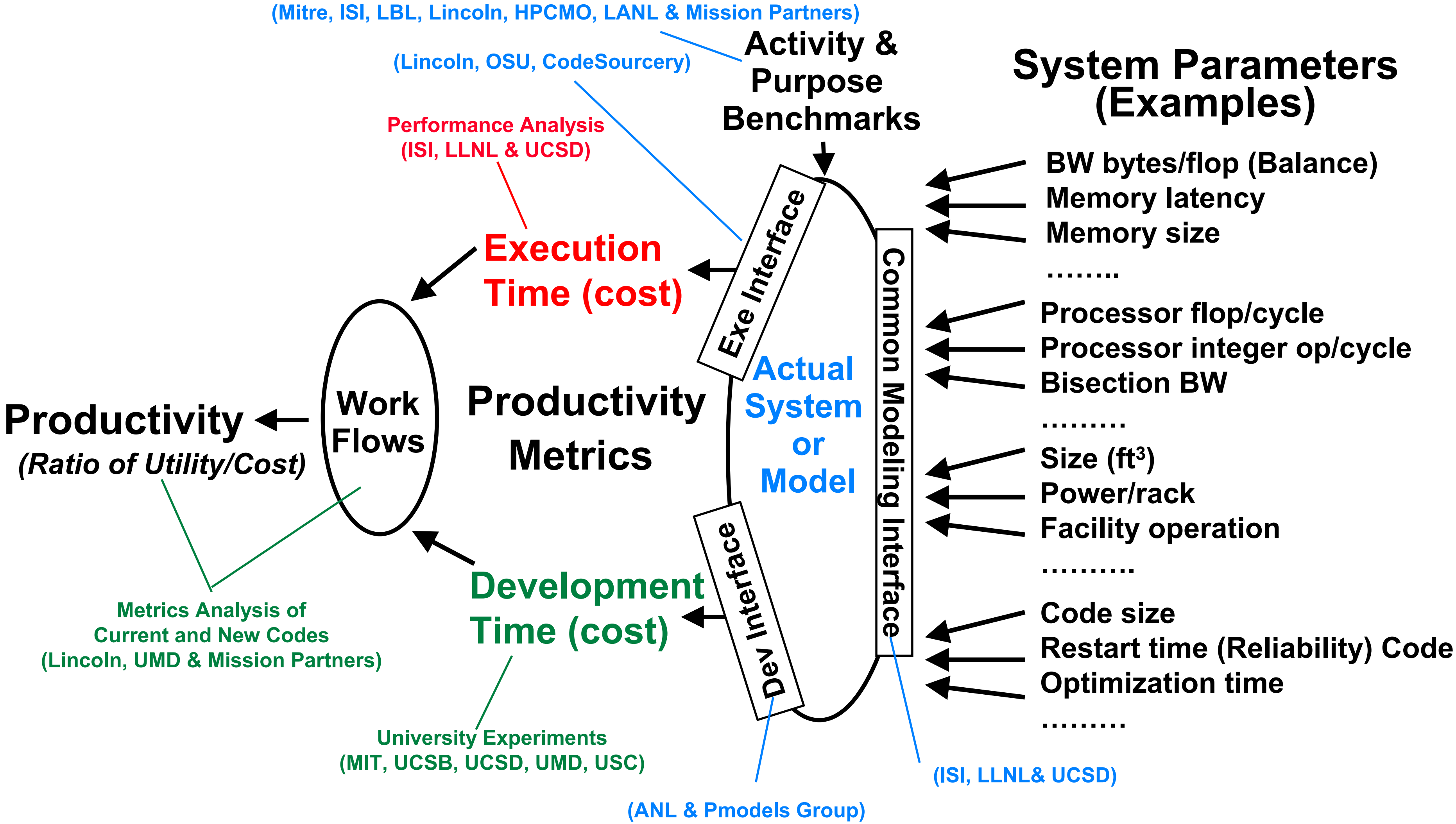
- Least common denominator standards
- Difficult to use
- Difficult to optimize

- **HPCS needs a validated assessment methodology that values the "right" vendor innovations**
- **Allow tradeoffs between Execution and Development Time**

Phase 1: Productivity Framework



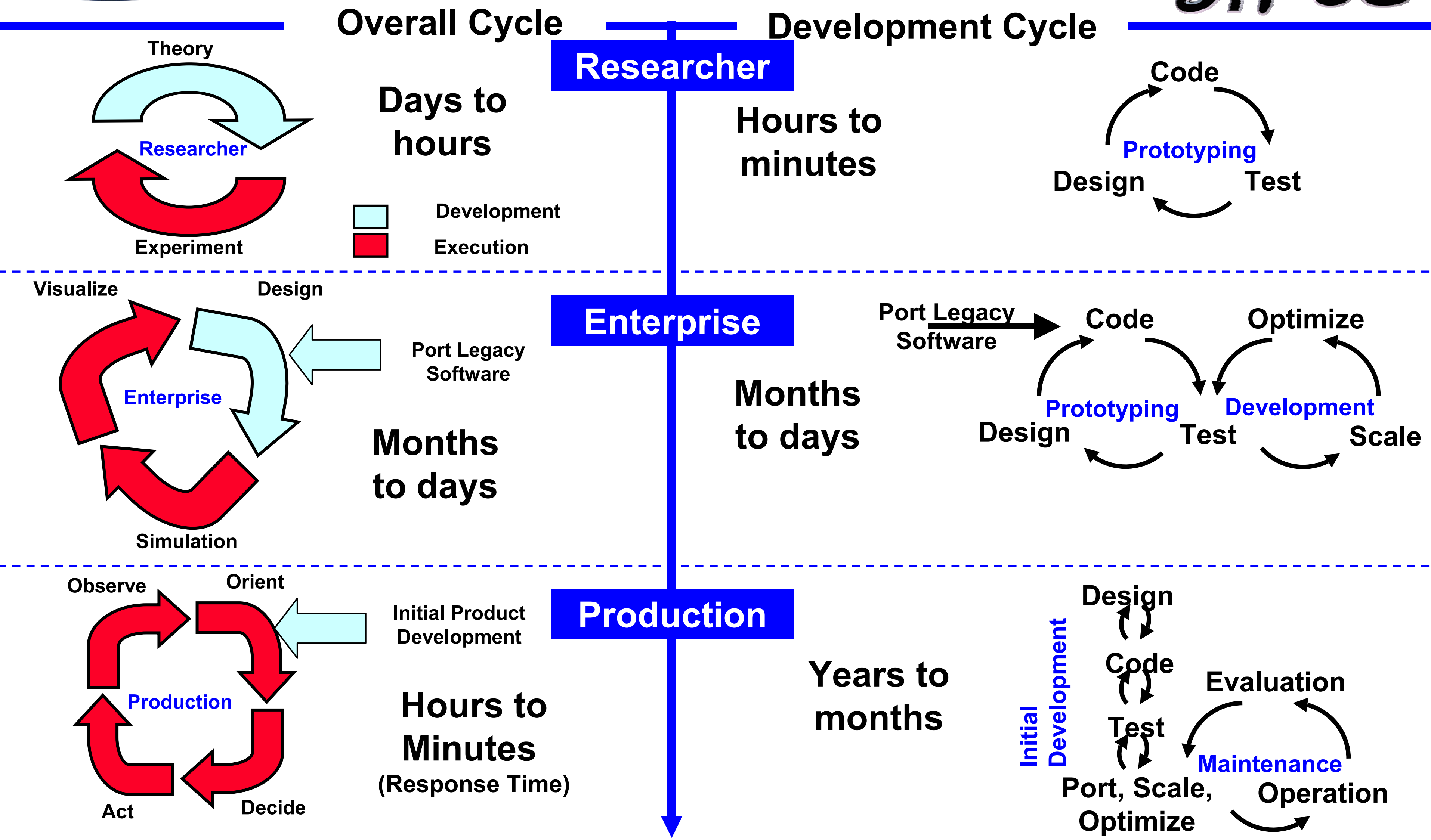
Phase 2: Implementation



Outline

- Introduction
- **Workflows**
  - *Lone Researcher*
  - *Enterprise*
  - *Production*
- Metrics
- Models & Benchmarks
- Schedule and Summary

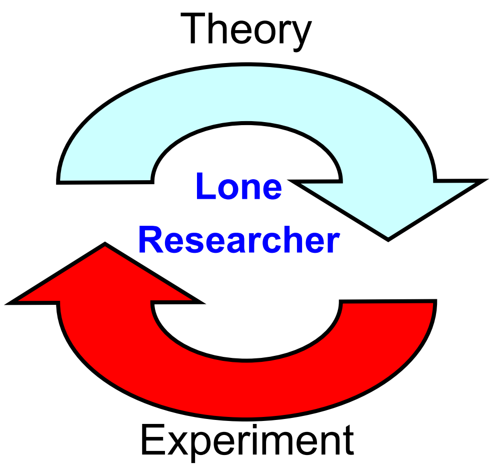
HPCS Mission Work Flows



HPCS Productivity Factors: Performance, Programmability, Portability, and Robustness are very closely coupled with each work flow

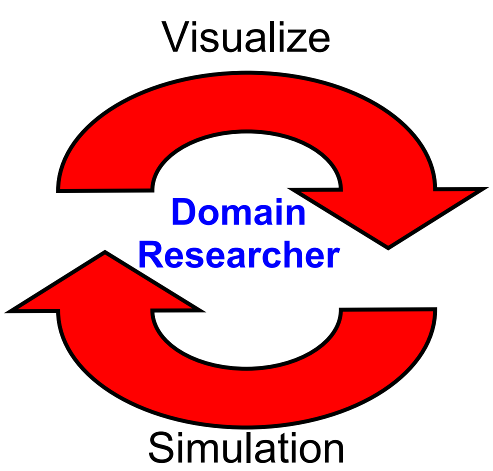
Lone Researcher

- 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 $\rightarrow$ 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



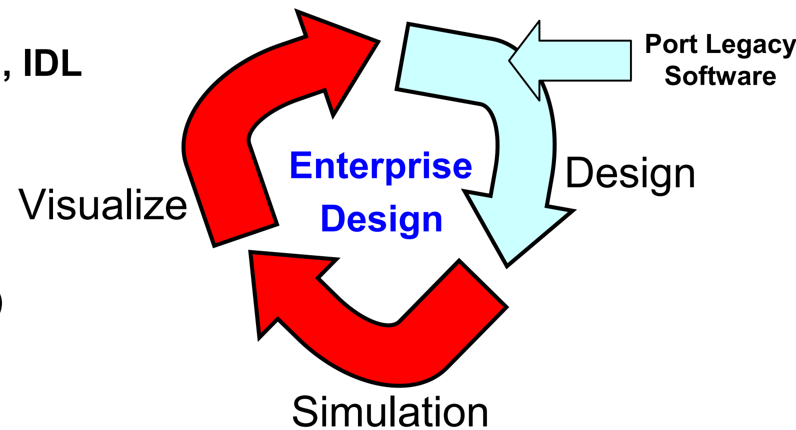
Enterprise Design

- **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**



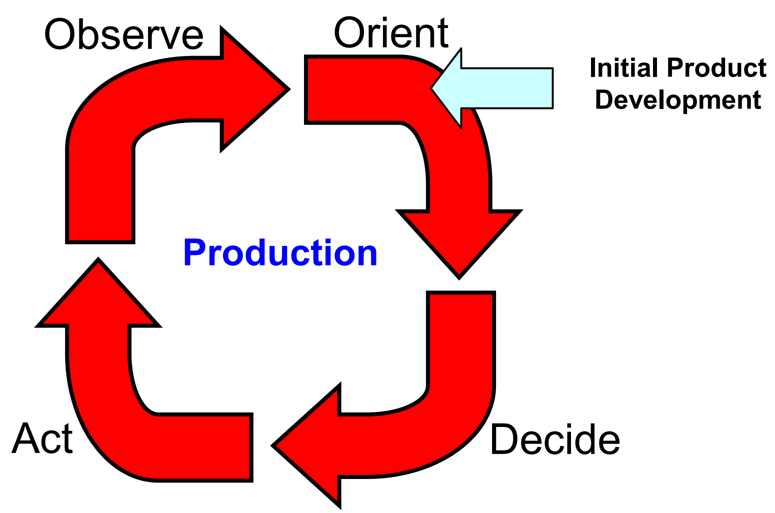
Production

- **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, $\rightarrow$ C++**

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



HPC Workflow SW Technologies

Production Workflow

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

Workstation			Supercomputer	
Algorithm Development	Spec	Design, Code, Test	Port, Scale, Optimize	Run

Operating Systems: Linux, RT Linux

Compilers: Matlab, Java, C++, OpenMP, F90, UPC, Coarray

Libraries: CORBA, VSIPL, ||VSIPL++, MPI, DRI, ATLAS, BLAS, FFTW, PETE, PAPI

Tools: UML, Globus, TotalView

Problem Solving Environments: CCA, ESMF, POOMA, PVL

Mainstream Software | HPC Software

Example: Coding vs. Testing

Workflow Breakdown (NASA SEL)

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

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?

Environment	Problem Size: Small (Workstation)	Medium (Cluster)	Full (HPC)
Prototype (Matlab)	X		
Serial (C/Fortran)	X		
Parallel (OpenMP)	X	X	X

New Result?
New Bug?

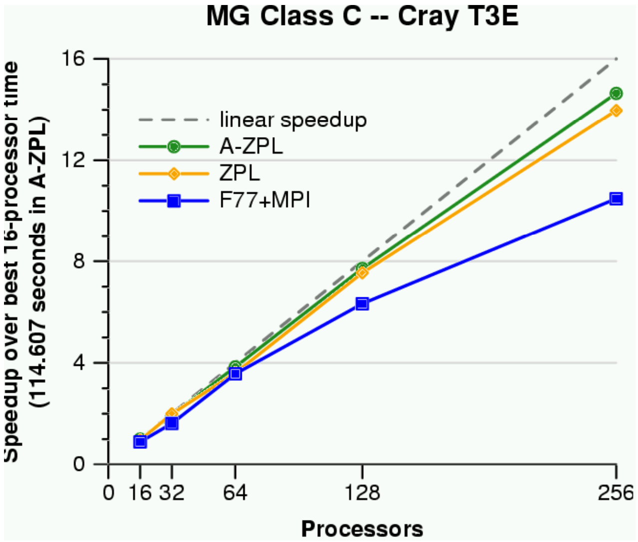
Outline

- Introduction
- Workflows
- **Metrics**
  - *Existing Metrics*
  - *Dev. Time Experiments*
  - *Novel Metrics*
- Models & Benchmarks
- Schedule and Summary

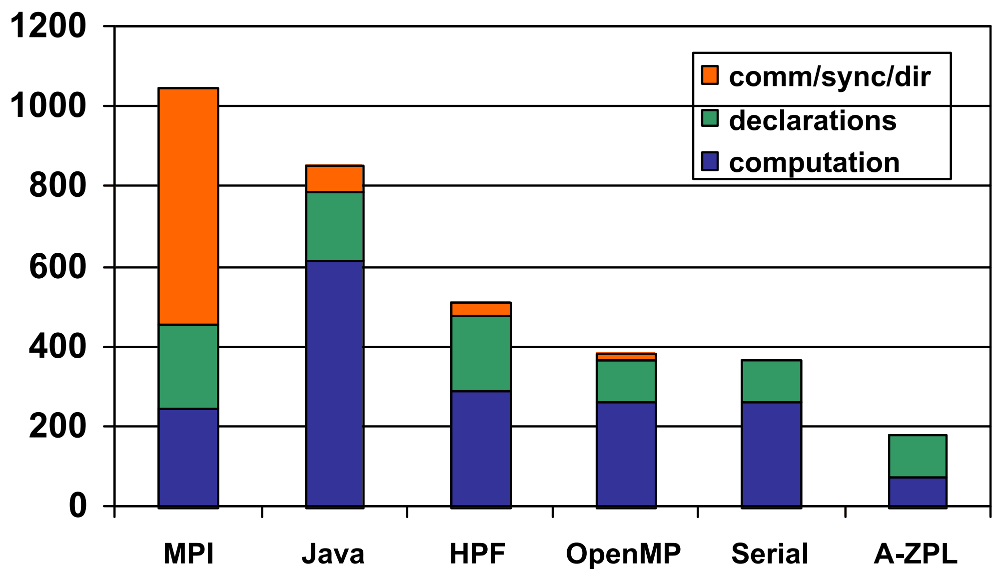
Example Existing Code Analysis

Analysis of existing codes used to test metrics and identify important trends in productivity and performance

MG Performance



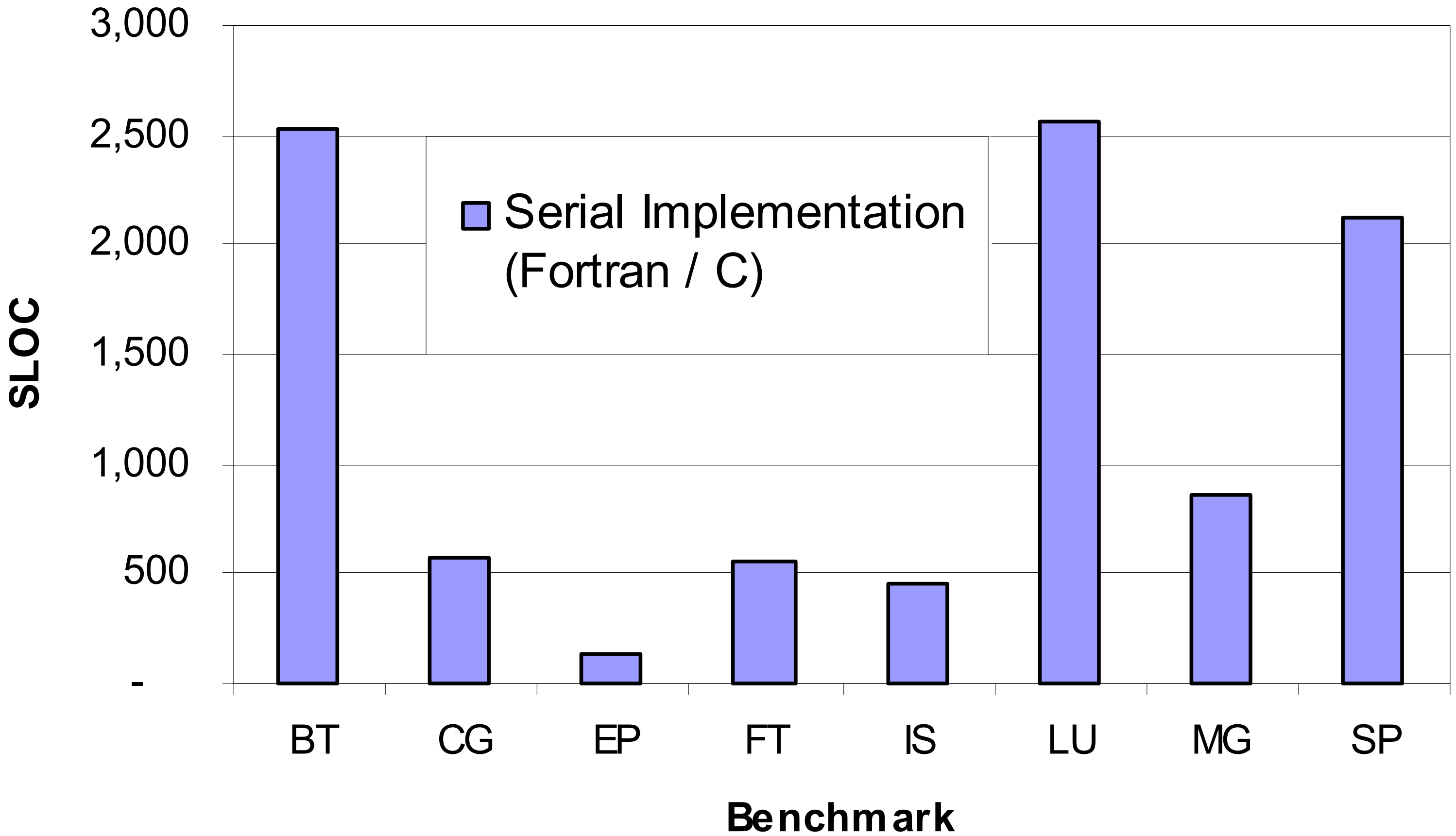
NAS MG Linecounts



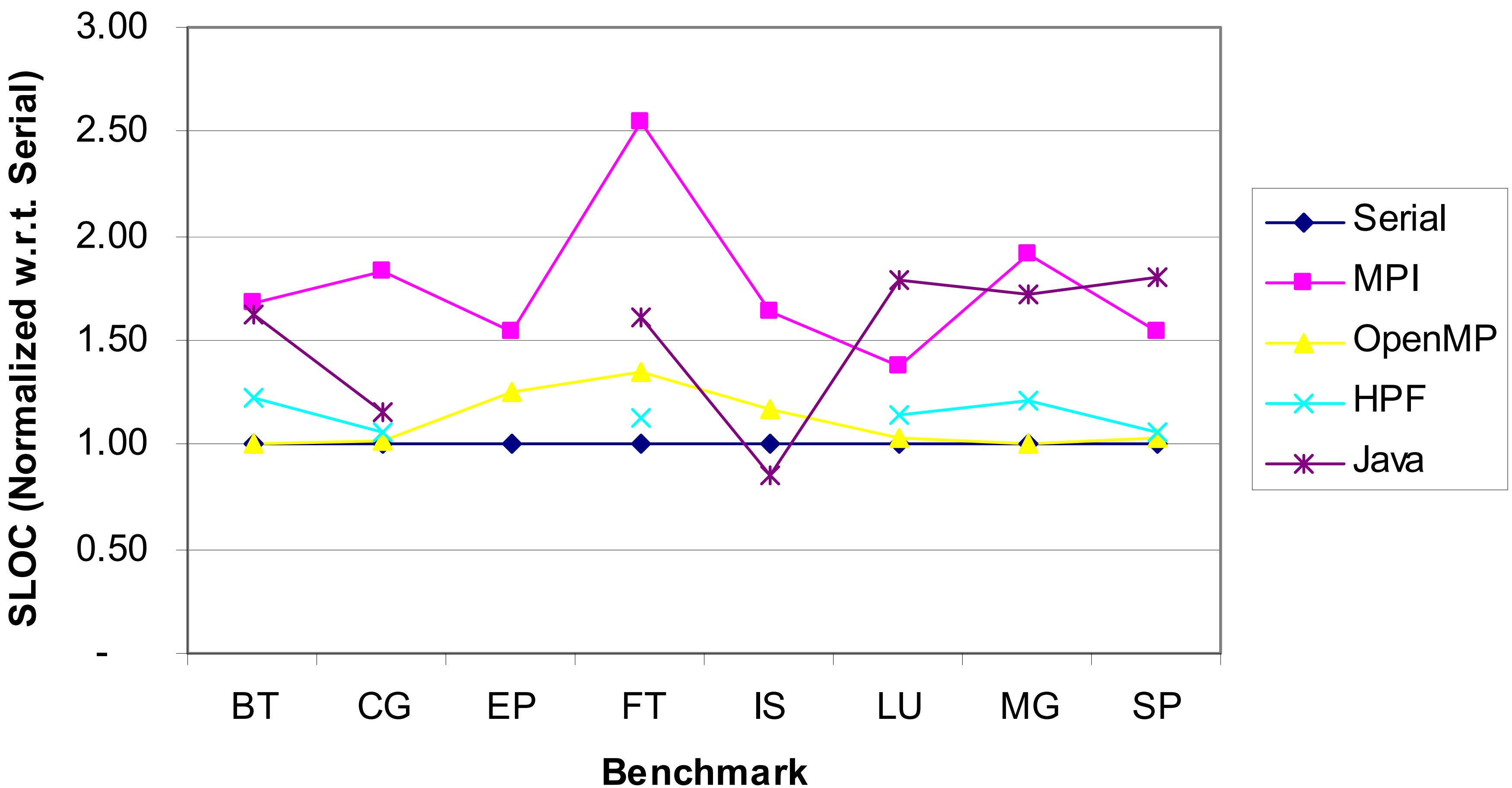
NPB Implementations

Benchmark	Languages							
	Serial Fortran	Serial C	Fortran / MPI	C / MPI	Fortan / OpenMP	C / OpenMP	HPF	Java
BT	■		■		■		■	■
CG	■		■		■		■	■
EP	■		■		■			
FT	■		■		■		■	■
IS		■		■		■		■
LU	■		■		■		■	■
MG	■		■		■		■	■
SP	■		■		■		■	■

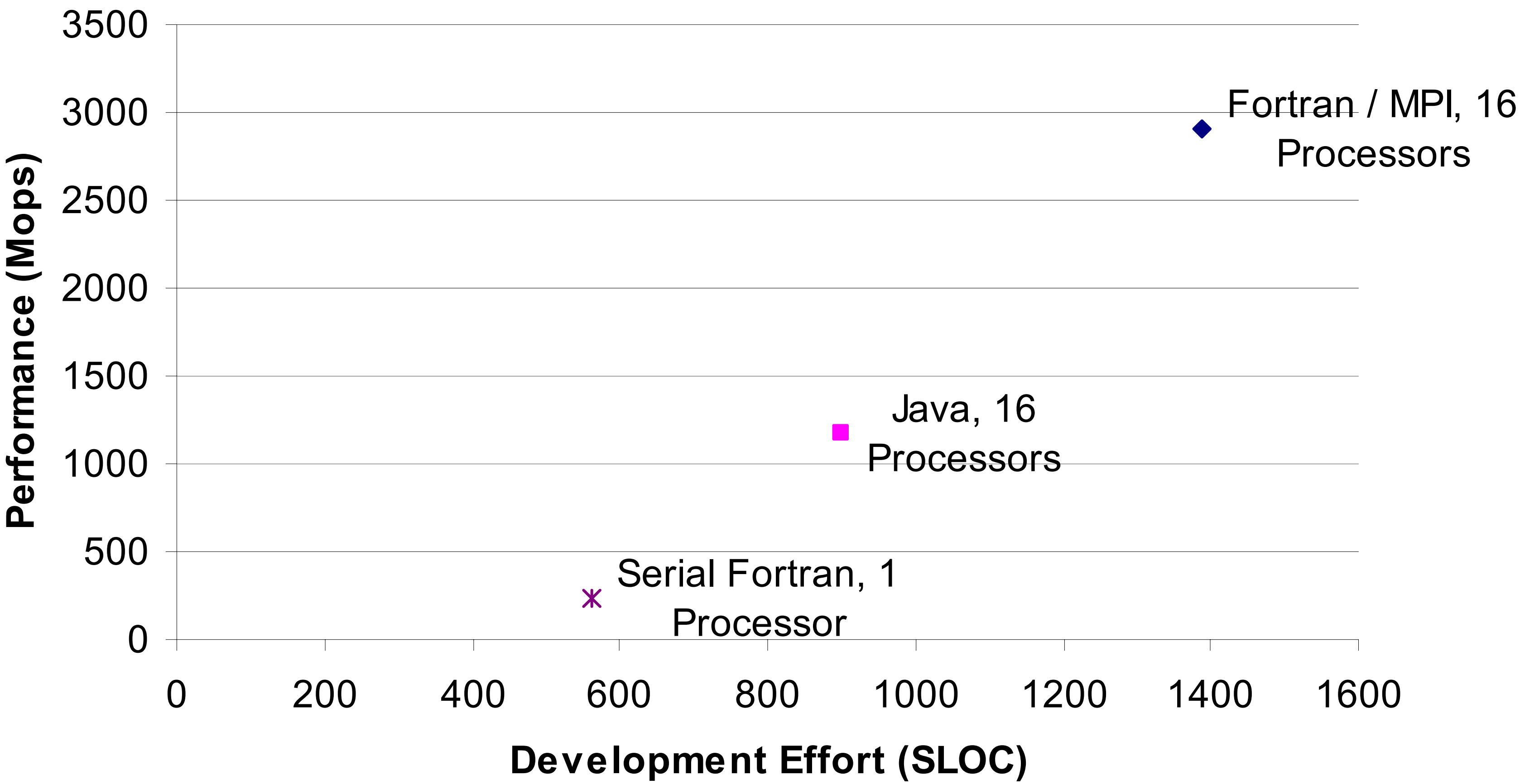
Source Lines of Code (SLOC) for the NAS Parallel Benchmarks (NPB)



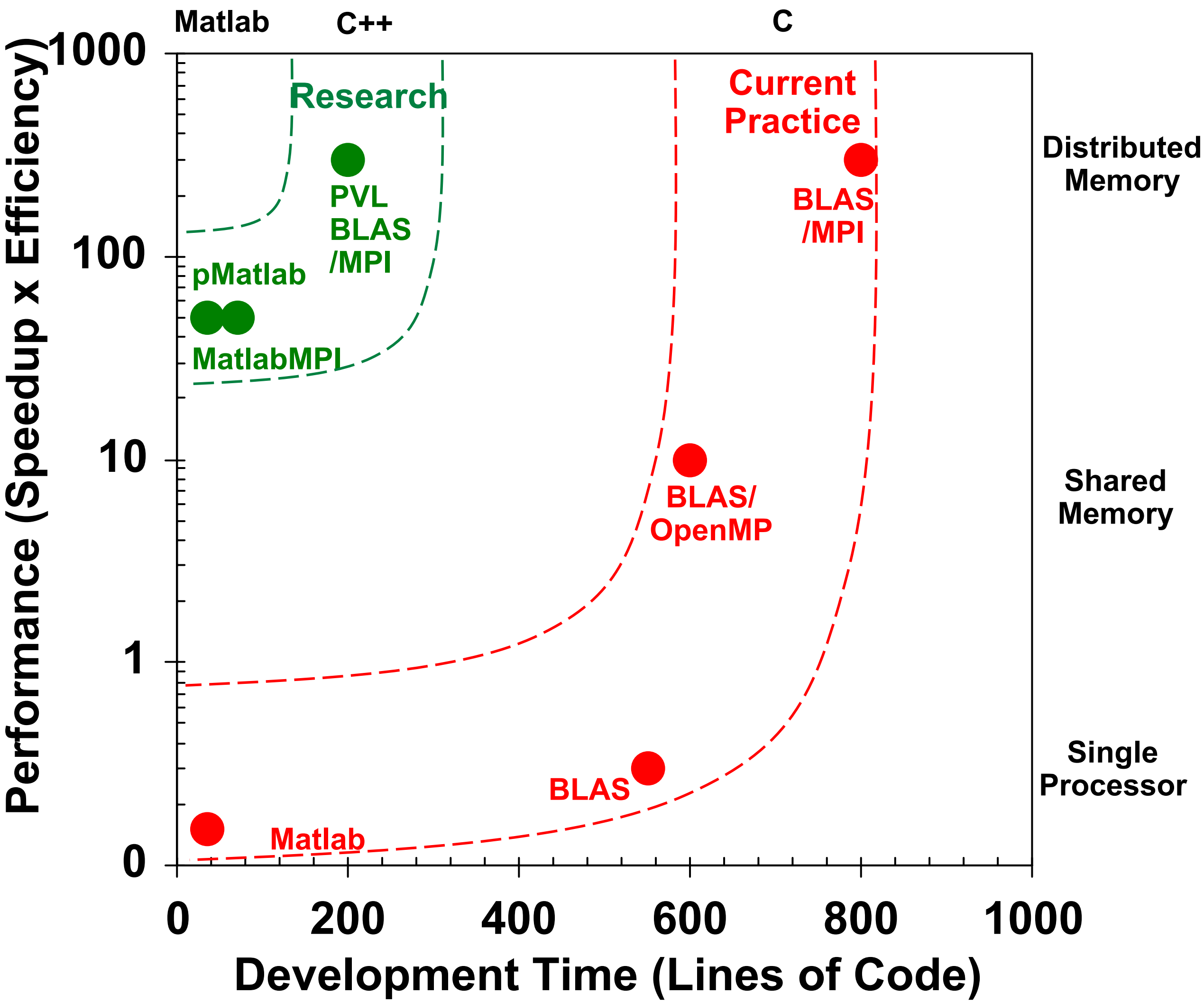
Normalized SLOC for All Implementations of the NPB



NAS FT Performance vs. SLOCs



Example Experiment Results (N=1)



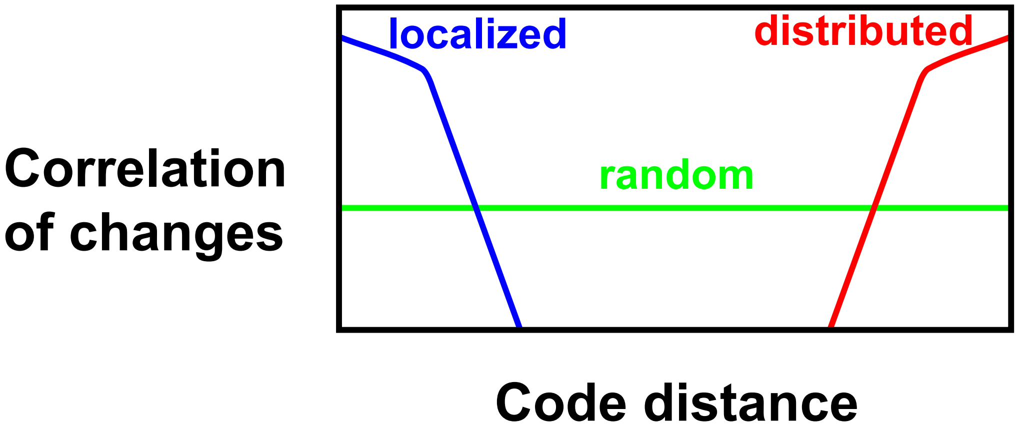
- Same application (image filtering)
- Same programmer
- Different langs/libs
  - •Matlab
  - •BLAS
  - •BLAS/OpenMP
  - •BLAS/MPI*
  - •PVL/BLAS/MPI*
  - •MatlabMPI
  - •pMatlab*

*Estimate

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 (Δx) to change performance (Δy)
  - 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)

$$\Psi_w = \frac{S_P \times E \times A}{c_f \times \{\Gamma \times (\bar{\rho} \bullet \bar{n})\} + (c_m + c_o) \times T}$$

Utility (Snir)

$$P(S,A,U(.)) = \min_{\cos t} \frac{U(T(S,A,Cost))}{Cost}$$

Productivity Factor Based (Kepner)

$$\text{productivity}_{\text{GUPS} \ldots \text{Linpack}} \approx \left(\frac{\left(\frac{\text{useful ops}}{\text{second}}\right)_{\text{GUPS} \ldots \text{Linpack}}}{\text{Hardware Cost}} \right) \left(\begin{matrix}\text{productivity}\\ \text{factor}\end{matrix}\right) \left(\begin{matrix}\text{mission}\\ \text{factor}\end{matrix}\right)$$

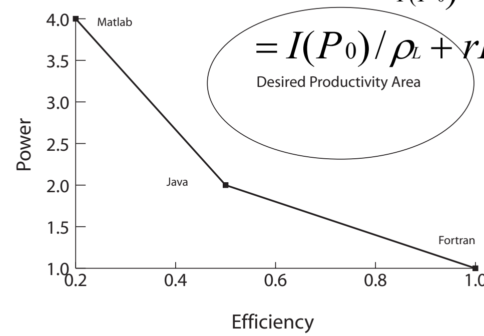
$$\left(\begin{matrix}\text{productivity}\\ \text{factor}\end{matrix}\right) \approx \left(\begin{matrix}\text{Language}\\ \text{Level}\end{matrix}\right) \times \left(\begin{matrix}\text{Parallel}\\ \text{Model}\end{matrix}\right) \times \text{Portability} \times \frac{\text{Availability}}{\text{Maintenance}}$$

Efficiency and Power (Kennedy, Koelbel, Schreiber)

$$T(P_L) = I(P_L) + rE(P_L)$$

$$= I(P_0) \cdot \frac{I(P_L)}{I(P_0)} + rE(P_0) \cdot \frac{E(P_L)}{E(P_0)}$$

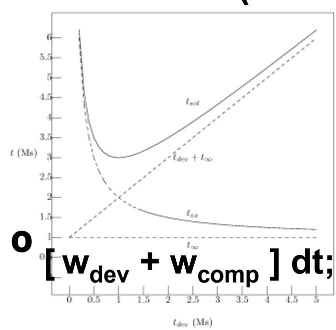
$$= I(P_0)/\rho_L + rE(P_0)/\varepsilon_L$$



CoCoMo II (software engineering community)

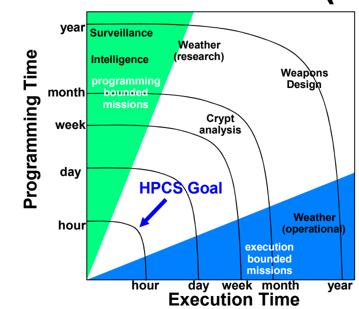
$$\left(\begin{matrix}\text{Effort}\\ \text{Multipliers}\end{matrix}\right) \times A \times (\text{Size})^{\left(\begin{matrix}\text{Scale}\\ \text{Factors}\end{matrix}\right)}$$

Least Action (Numrich)



$$S = \int [w_{dev} + w_{comp}]\, dt; \qquad \delta S = 0$$

Time-To-Solution (Kogge)



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

Lines of code
Function Points
Reuse
Re-engineering
Maintenance

Code Size = (New) + (Reused) + (Re-engineered) + (Maintained)

Measured in lines of code or functions points (converted to lines of code)

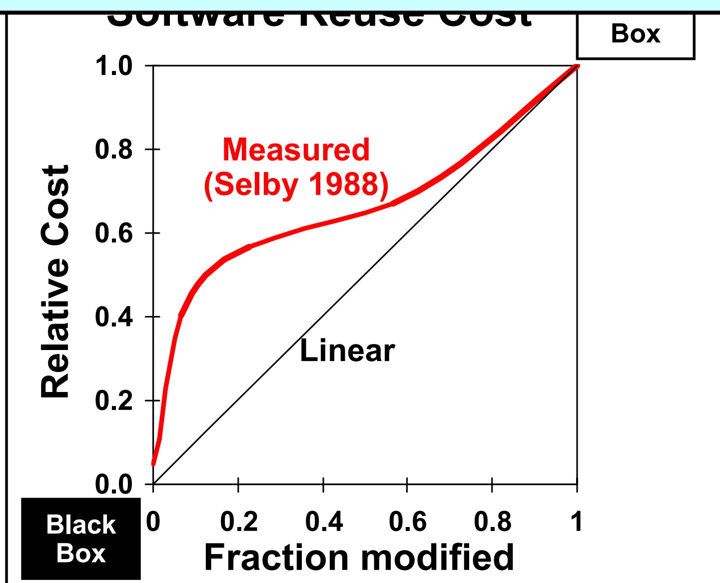
Lines per function point	
C, Fortran	~100
Fortran77	~100
C++	~30
Java	~30
Matlab	~10
Python	~10
Spreadsheet	~5

HPC Challenge Areas

Function Points
High productivity languages not available on HPC

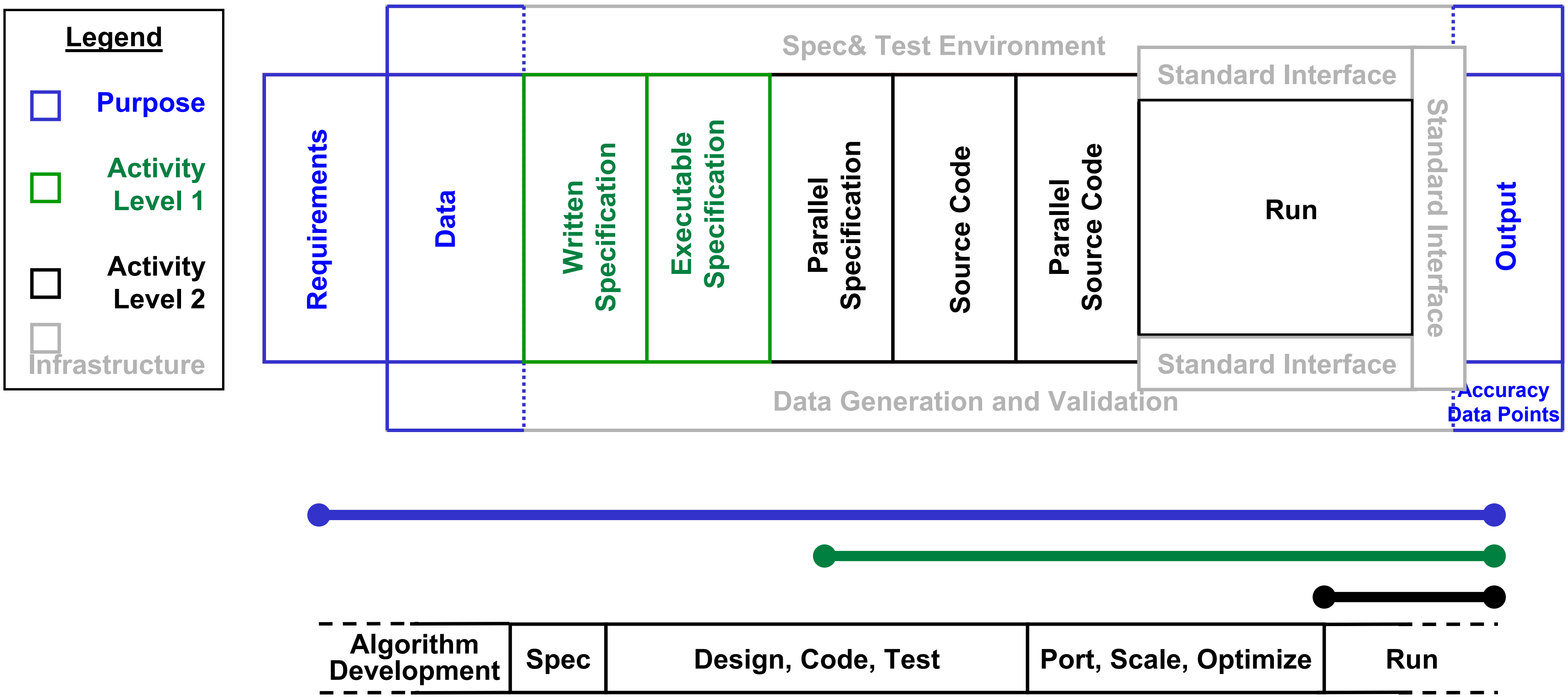
Reuse
Nonlinear reuse effects. Performance requirements dictate "white box" reuse model

- 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

Activity & Purpose Benchmark



Development Workflow

Activity 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

Mission Area	Kernels	Application	Source
Stockpile Stewardship	Random Memory Access	UMT2000	ASCI Purple Benchmarks
	Unstructured Grids		
	Eulerian Hydrocode	SAGE3D	ASCI Purple Benchmarks
	Adaptive Mesh		
	Unstructured Finite Element Model	ALEGRA	Sandia National Labs
	Adaptive Mesh Refinement		
Operational Weather and Ocean Forecasting	Finite Difference Model	NLOM	DoD HPCMP TI-03
Army Future Combat Weapons Systems	Finite Difference Model	CTH	DoD HPCMP TI-03
	Adaptive Mesh Refinement		
Crashworthiness Simulations	Multiphysics Nonlinear Finite Element	LS-DYNA	Available to Vendors

Other Kernels	Lower / Upper Triangular Matrix Decomposition	LINPACK	Available on Web
	Conjugate Gradient Solver		DoD HPCMP TI-03
	QR Decomposition		Paper & Pencil for Kernels
	1D FFT		Paper & Pencil for Kernels
	2D FFT		Paper & Pencil for Kernels
	Table Toy (GUP/s)		Paper & Pencil for Kernels
	Multiple Precision Mathematics		Paper & Pencil for Kernels
	Dynamic Programming		Paper & Pencil for Kernels
	Matrix Transpose [Binary manipulation]		Paper & Pencil for Kernels
	Integer Sort [With large multiword key]		Paper & Pencil for Kernels
	Binary Equation Solution		Paper & Pencil for Kernels
	Graph Extraction (Breadth First) Search		Paper & Pencil for Kernels
	Sort a large set		Paper & Pencil for Kernels
	Construct a relationship graph based on proximity		Paper & Pencil for Kernels
	Various Convolutions		Paper & Pencil for Kernels
	Various Coordinate Transforms		Paper & Pencil for Kernels
	Various Block Data Transfers		Paper & Pencil for Kernels

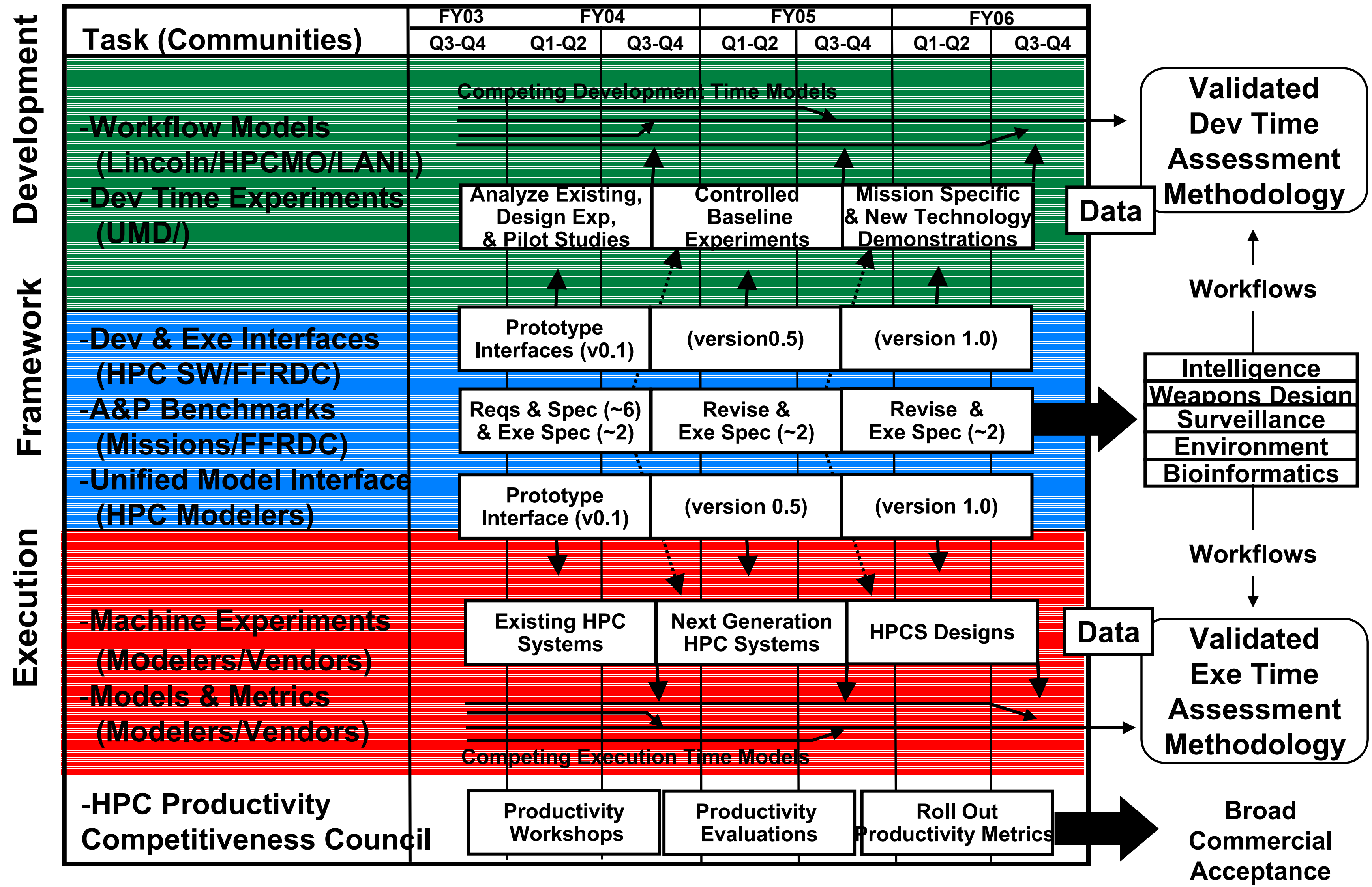
Bio-Application	Kernels	Application	Source
Quantum and Molecular Mechanics	Macromolecular Dynamics	CHARMM	http://yuri.harvard.edu/
	Energy Minimization		
	MonteCarlo Simulation		
Whole Genome Analysis	Sequence Comparison	Needleman-Wunsch	http://www.med.nyu.edu/rcr/rcr/course/sim-sw.html
		BLAST	http://www.ncbi.nlm.nih.gov/BLAST/
		FASTA	http://www.ebi.ac.uk/fasta33/
		HMMR	http://hmmer.wustl.edu/
Systems Biology	Functional Genomics	BioSpice (Arkin, 2001)	http://genomics.lbl.gov/~aparkin/Group/Codebase.html
	Biological Pathway Analysis		

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



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:

PI: Elnozahy   PI: Gustafson   PI: Smith

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

PI: Kepner   PI: Lucas   PI: Basili   PI: Benson & Snavely

PI: Koester   PIs: Vetter, Lusk, Post, Bailey   PIs: Gilbert, Edelman, Ahalt, Mitchell

Goal:

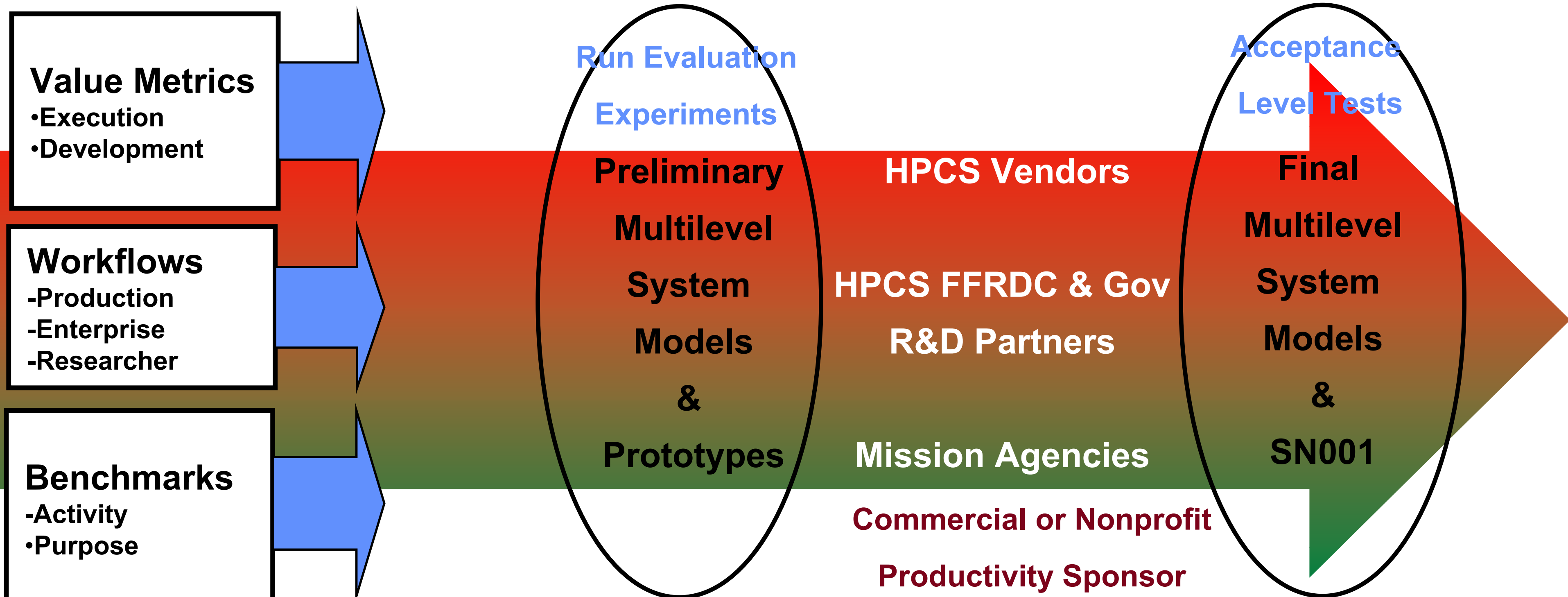
- Develop a procurement quality assessment methodology that will be the basis of 2010+ HPC procurements

Productivity Framework Overview

Phase I: Define Framework & Scope Petascale Requirements

Phase II: Implement Framework & Perform Design Assessments

Phase III: Transition To HPC Procurement Quality Framework



HPCS needs to develop a procurement quality assessment methodology that will be the basis of 2010+ HPC procurements