HPCS Application Analysis and Assessment

- Phase 1 Summary -

David Koester / MITRE
Jeremy Kepner / MIT Lincoln Laboratory

HPC User Forum
Sundance, Utah

Outline

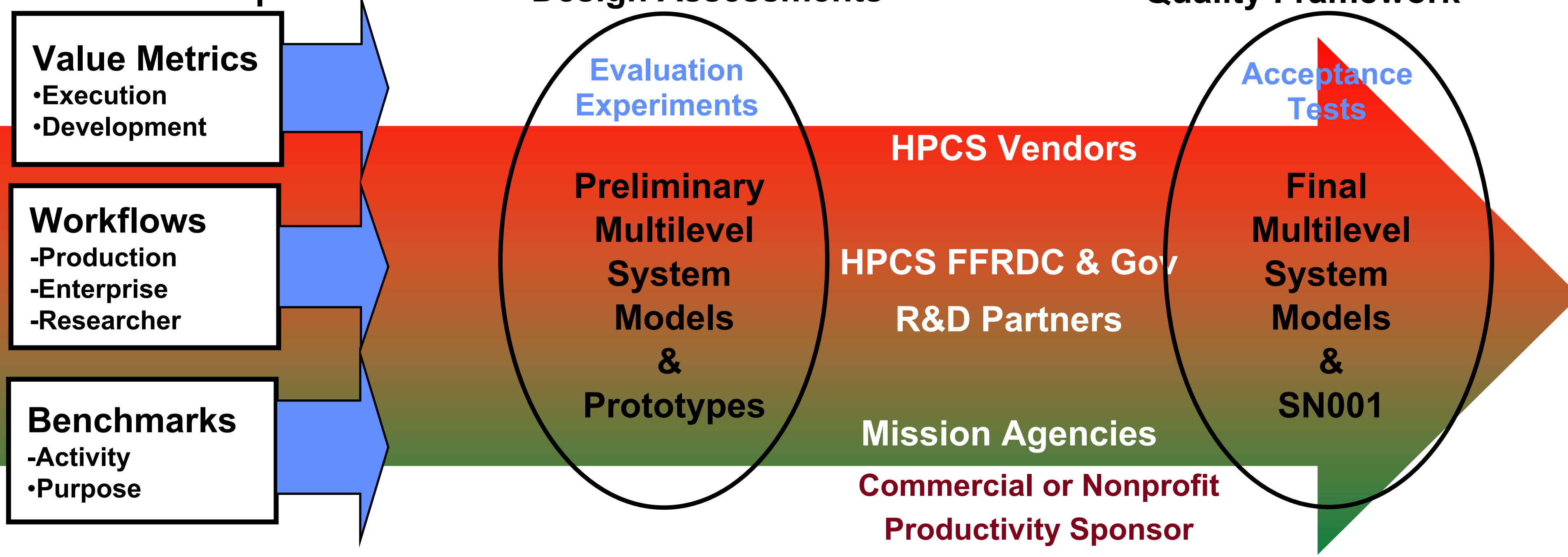
- **Assessment and Metrics**
  - ***Overview***
  - ***Process Flow***
  - ***Assessment Framework***
  - ***Defining Productivity***
- **Workflows**
- **Benchmarks**
- **Continuing Challenges**
- **Summary**

Productivity Framework Overview

Phase I: Define Framework & Scope Petascale Requirements

Phase II: Implement Framework & Perform Design Assessments

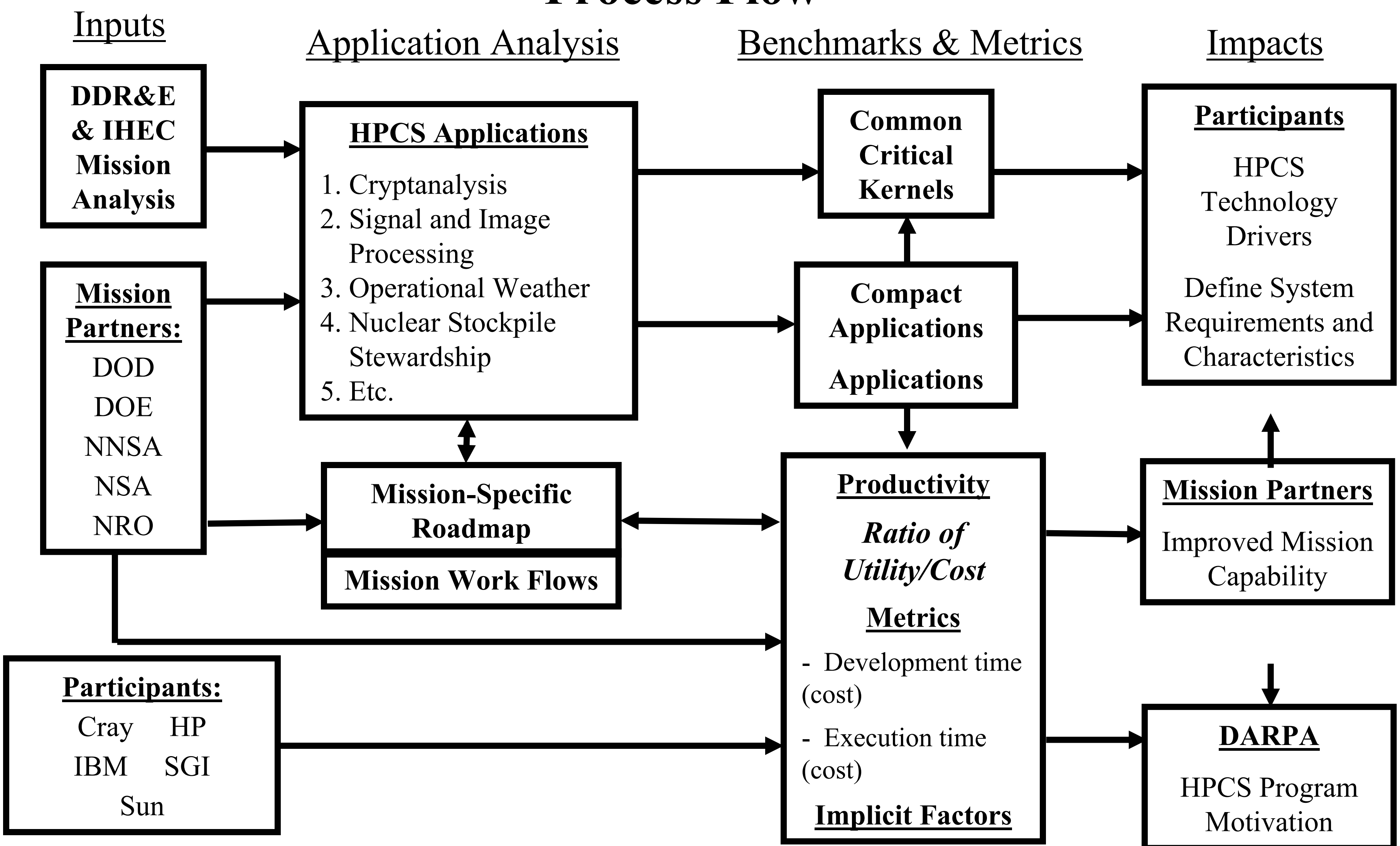
Phase III: Transition To HPC Procurement Quality Framework



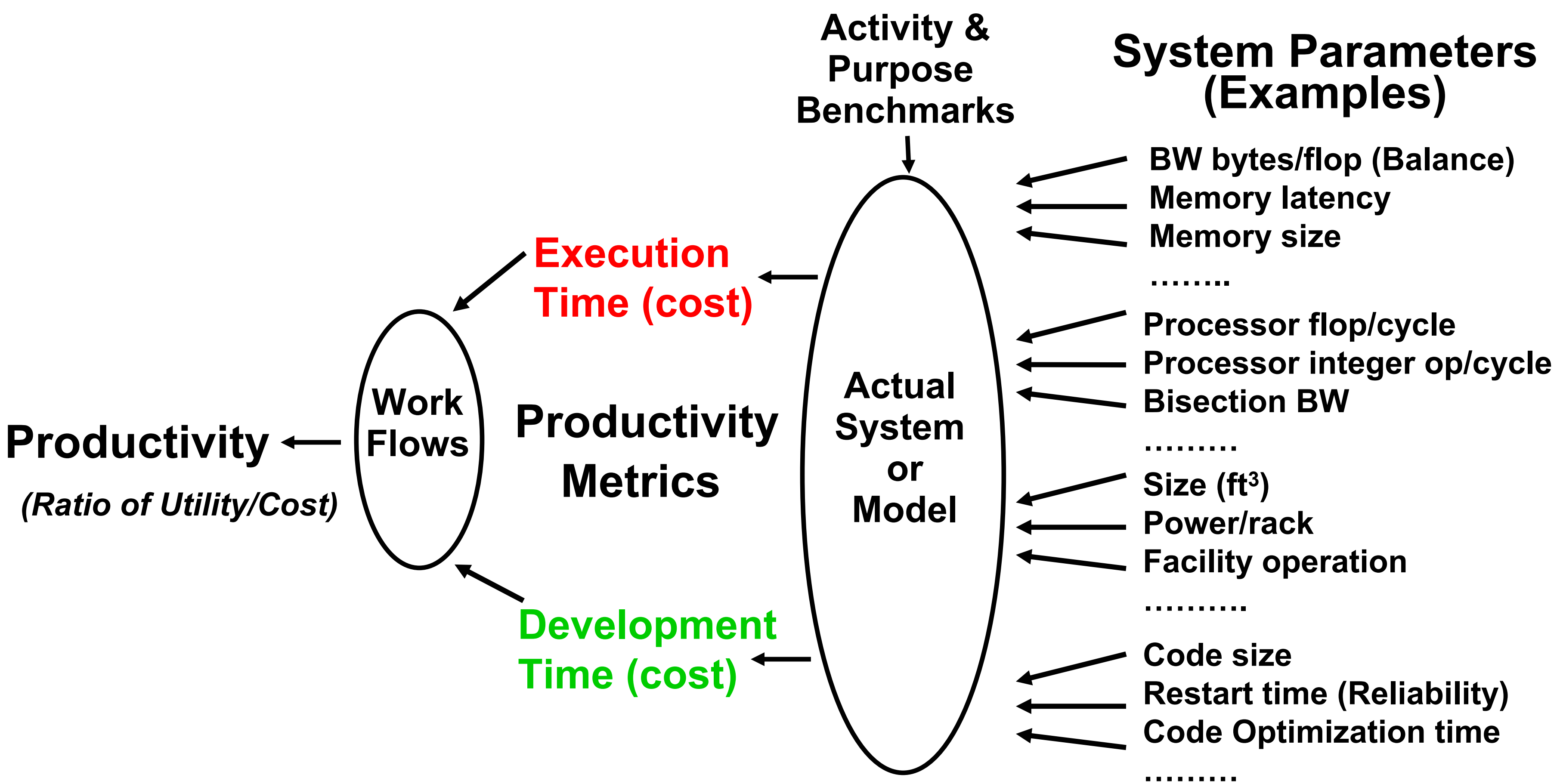
- **Program continuously integrates mission and vendor input**
  - Enables vendors to perform design assessments and measure HPCS objectives progress
  - Enables mission partners and program management to understand vendor designs via scaled models/tools using vendor supplied parameters

Application Analysis/ Performance Assessment

Process Flow



HPCS Assessment Framework



HPCS Productivity Factors:
Performance, Programmability, Portability, and Robustness

HPC Productivity: A New Era

A New HPC Sub-discipline

Special Model with Work Estimator (Sterling)

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

Utility (Snir)

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

Productivity Factor Based (Kepner)

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

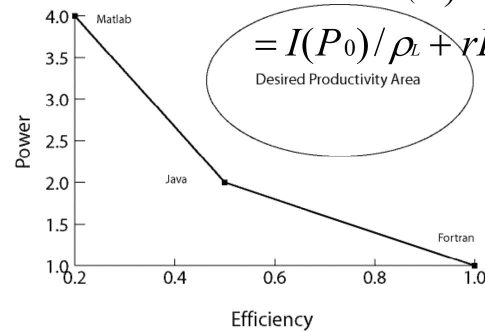
$$\left(\begin{array}{c}\text{productivity}\\ \text{factor}\end{array}\right) \approx \left(\begin{array}{c}\text{Language}\\ \text{Level}\end{array}\right) \times \left(\begin{array}{c}\text{Parallel}\\ \text{Model}\end{array}\right) \times \text{Portability} \times \frac{\text{Availability}}{\text{Maintenance}}$$

Efficiency and Power (Kennedy, Koelbel, Schreiber)

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

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

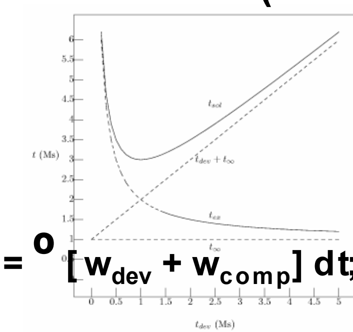
$$= I(P_0)/\rho_L + rE(P_0)/\varepsilon_L$$



CoCoMo II (software engineering community)

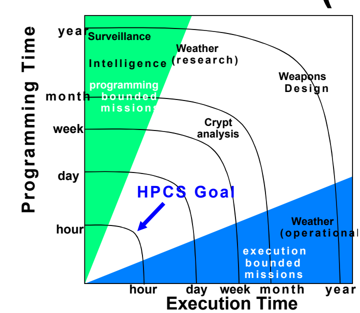
$$\left(\begin{array}{c}\text{Effort}\\ \text{Multipliers}\end{array}\right) \times A \times \left(\text{Size}\right)^{\left(\begin{array}{c}\text{Scale}\\ \text{Factors}\end{array}\right)}$$

Least Action (Numrich)



$$S = \int_0 \left[w_{dev} + w_{comp} \right] dt; \qquad \delta S = 0$$

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 (watch for SC 2003 Panel/BoF; International Journal of High Performance Computing Applications - Special Issue)

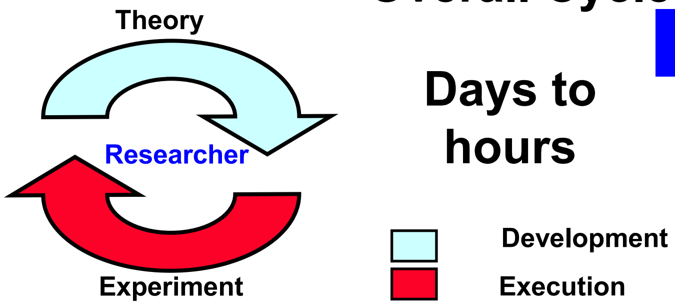
Outline

- Assessment and Metrics
- **Workflows**
  - *Overview*
  - *Definitions*
  - *Common Challenges*
- Benchmarks
- Summary

HPCS Mission Work Flows

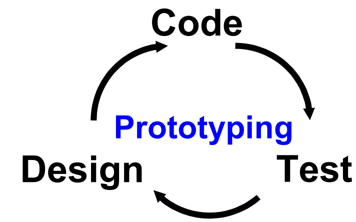
Overall Cycle / Development Cycle

Researcher

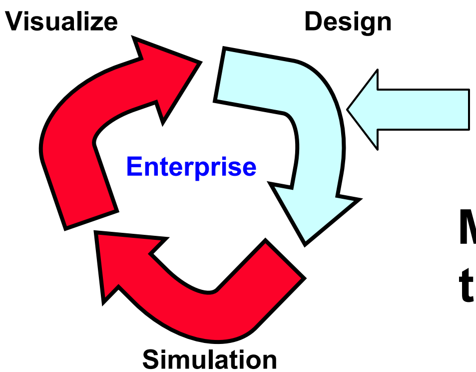


Days to hours

Hours to minutes



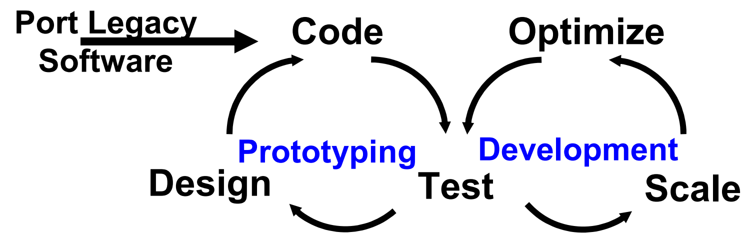
Enterprise



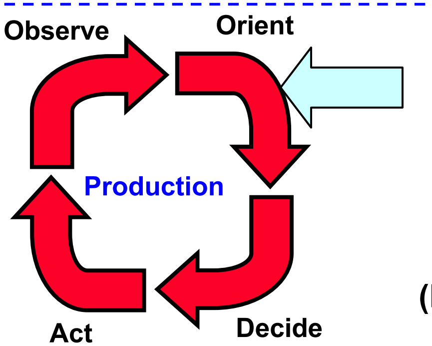
Port Legacy Software

Months to days

Months to days



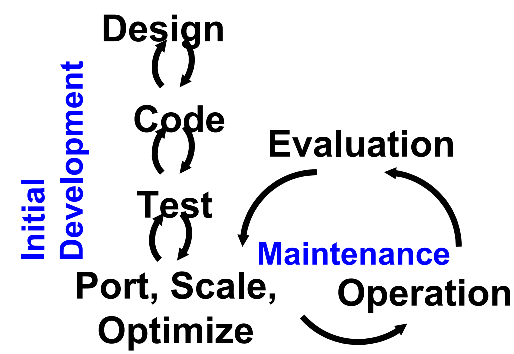
Production



Initial Product Development

Hours to Minutes (Response Time)

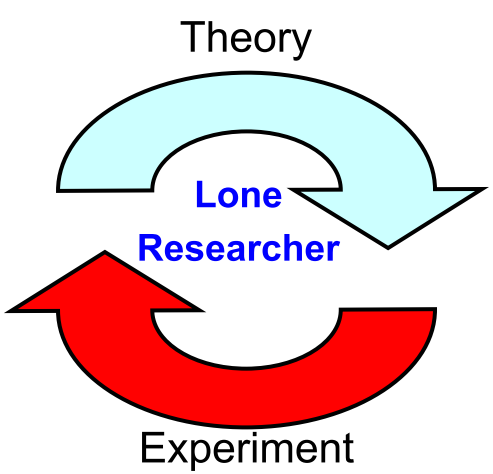
Years to months



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



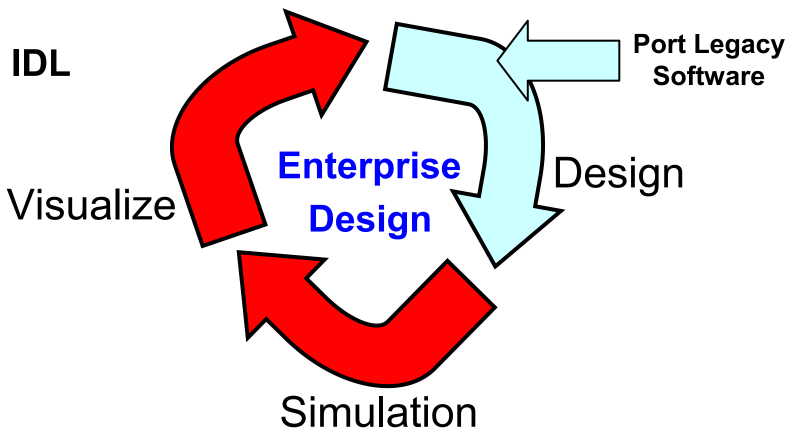
Enterprise Design

- 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

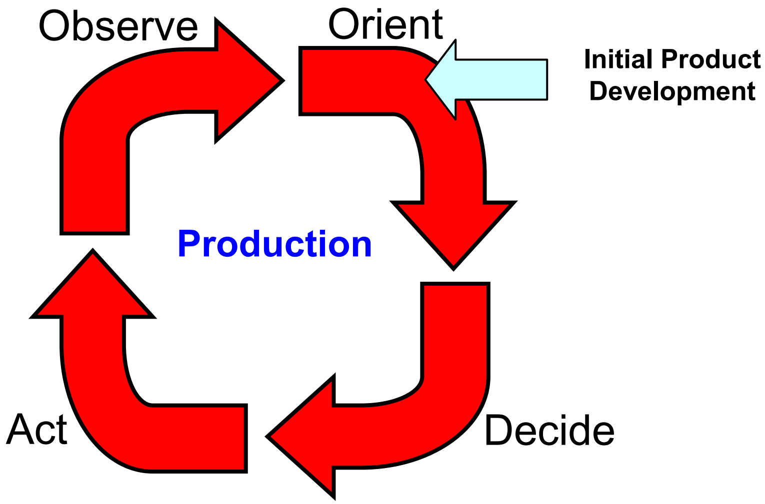
- Environment overview
  - Duration: ~1 year   Team size: 2-20
  - Machines: workstations, clusters, hpc
  - Languages: FORTRAN, C, $\rightarrow$ C++, Matlab, Python, IDL
  - Libraries: open math and communication libraries

- 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



Common Development Challenges

- **Workstations are dominant development platform**
  - Scaling from workstations to clusters to HPC is difficult
  - Special hardware features (SIMD, DMA, …) are avoided
  - Need transparent portability that preserves performance

- **Code reuse is essential**
  - Frameworks commonly employed for functional reuse, but
    - No formal application programmer interface (API)
    - Serial (difficult to make parallel)
    - Development and production are different
  - Need mission specific software frameworks that span
    - Development and production
    - Workstations, clusters, HPC+special hardware

- **Increased use of high level languages**
  - Preferred by domain experts, not software engineers
  - Limited availability on HPCs
  - Not high performance

- **A new approach: development code is HPC production quality?**

Outline

- Assessment and Metrics
- Workflows
- **Benchmarks**
  - *Scope*
  - *Relationships*
  - *Learning from History*
  - *Credible System Performance*
  - *Interrelationships*
- Summary

HPCS Phase 1 Kernel and Application *Scope* Benchmarks

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		

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	BLAST	http://www.ncbi.nlm.nih.gov/BLAST/
Systems Biology	Functional Genomics	BioSpice (Arkin, 2001)	http://genomics.lbl.gov/~aparkin/Group/Codebase.html
	Biological Pathway Analysis		

Benchmark Relationships

	Fixed Size	Scalable
Activity Based (Well Suited for Execution Measurement)	**LINPACK** (Dongarra's performance.ps) **NAS Parallel** **SPEC HPC2002** ***HPCS Activity Applications***	**LINPEAK** (Top500) **Streams, Table Toy** ***HPCS Activity Kernels***
Purpose Based (Ideal for Development Measurement)	**"Discrete Math"** **Many RFP Suites**	**TPC-x, ECPerf** ***HPCS Purpose Suite***

HPCS Focus

HPCS Phase 1 – Scope Benchmarks
HPCS Phase 2 – Activity and Purpose Benchmarks

Learning from History

1990s HPC technology producers: Alliant, Cray Computer, Supercomputing Systems, Thinking Machines, Kendall Square Research, …

High Performance Computing Challenges for Future Systems

Demonstrate credible performance

"users can develop programs of infinite variety, and many types of programs lead to disastrous performance degradation on any particular system"

- Demonstrate (not claim) benefits across all mission areas
- Community is actively engage metrics development

Greatest grand challenge: *practical* parallelism (i.e. time-to-solution)

"solve the problem of designing practical parallel systems so that we will be able, forevermore, to improve computer performance through practical parallelism"

- Extract parallel performance without heroic programming efforts

David J Kuck, 1996

HPCS: Mission Decomposition DoD HPCMP Resource Center

CTAs†
- CSM
- CFD
- CCM
- CEA
- CWO
- SIP
- FMS
- EQM
- CEN
- IMT

Applications‡
- NASTRAN
- FAST3D
- LS-DYNA3D
- COBALT
- FEFLO
- TBMD
- FMD
- MACH3
- SAR
- ⋮

Kernels

Computation:
- Finite Element
- Finite Volume
- 1D FFTs
- 2D FFTs
- Linear Solvers
- Matrix Multiply
- Dot Product
- SVD
- Pattern Matching
- Database Ops
- ⋮

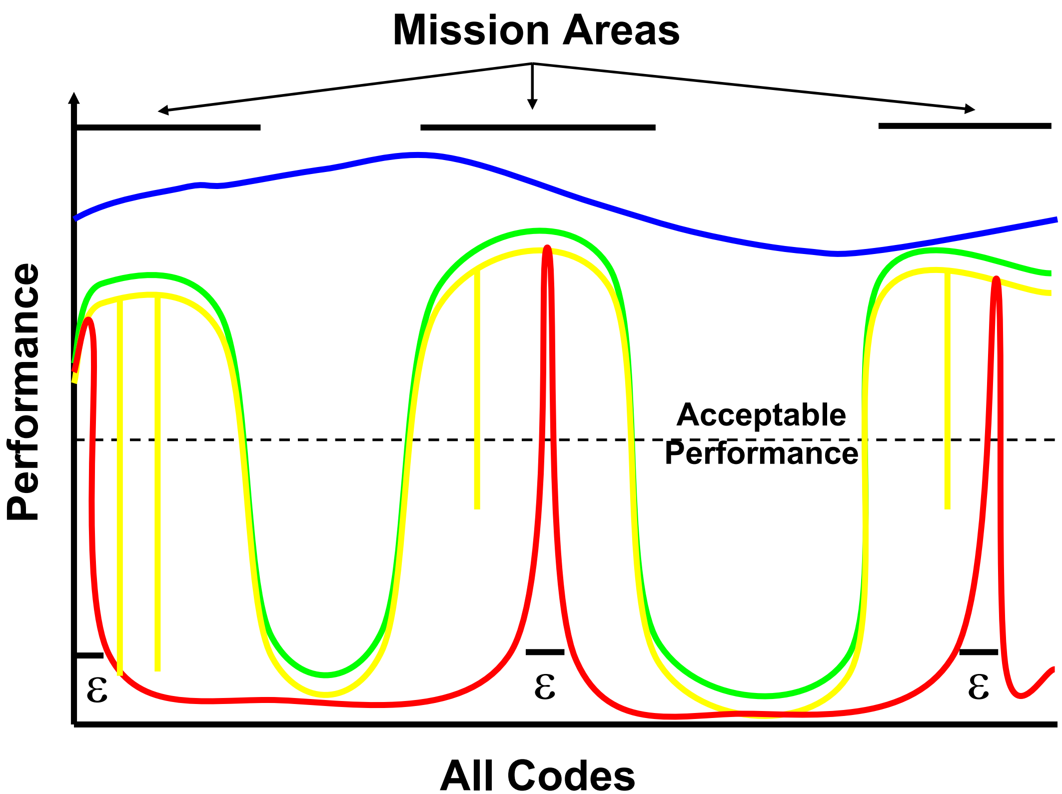
Communication:
- Point-to-Point
- Multicast
- Scatter/Gather
- Reductions
- ⋮

Architecture
- Operations
- Local Memory
- Global Memory
- Input/Output
- ⋮

HPCS needs to provide credible performance across all applications that are run at a DoD HPCMP Resource Center

†http://www.hpcmo.hpc.mil/Htdocs/CHSSI/cta_description.html
‡http://www.hpcmo.hpc.mil/Htdocs/CHSSI/cta_projects.html

Credible System Performance Across a Mission Area



Universal (vision)
  all codes acceptable
Ensemble (goal)
  mission area acceptable
Ensemble w/exceptions (achievable)
  Ensemble but with exceptions
Existential (current practice)
  small number acceptable, but unstable

(Reference: David Kuck)

- Acceptable performance across an entire mission area
  - mission area ↔ all applications for a mission partner
- Current computing systems are unstable
  - small (ε) code change can produce a large decrease in performance
  - some applications exhibit acceptable performance, many don't

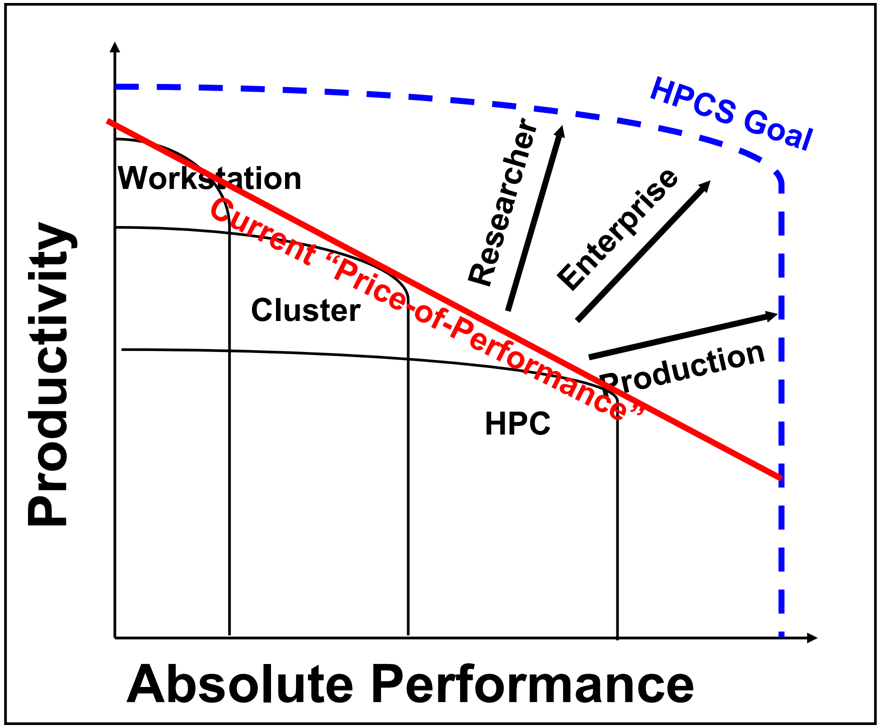
Interrelationships

Workflow	Perf.	Prog.	Port.	Robust.
		Productivity Factors		
Researcher		high		
Enterprise	high	high	high	high
Production	high			high

Mission Needs

System Requirements

- Workflows define scope of customer priorities
- Activity and Purpose benchmarks will be used to measure Productivity
- HPCS Goal is to add value to each workflow
  - Increase productivity while increasing performance



Summary

- Assessment and Metrics
  - Initial framework consisting of
    - Productivity Metrics (e.g. development time and execution time)
    - System Parameters (e.g. bandwidth, flops/cycle, size, power, lines-of-code, …)
    - Productivity Factors (performance, programmability, portability and robustness)
  - Ground breaking activity in understanding HPC productivity

- Workflows
  - Lone Researcher, Enterprise Development and Production with different mission and development cycles
  - Several common productivity challenges
    - Workstations for development; Code reuse; High level languages

- Benchmarks
  - Defines scope of applications of interest
  - Targets different aspects of workflow (activity vs. process)
  - Goal is performance across mission areas