Working Group 6
Performance Modeling, Metrics and Specifications

Chair: David Bailey

Vice Chair: Allen Snavely

WG6 – Performance Modeling, Metrics, and Specifications Charter

- Charter
  - Establish objectives of future performance metrics and measurement techniques to characterize system value and productivity to users and institutions. Identify strategies for evaluation including benchmarking of existing and proposed systems in support of user applications. Determine parameters for specification of system attributes and properties.
- Chair
  - David Bailey, Lawrence Berkeley National Laboratory
- Vice-Chair
  - Allan Snavely, UC San Diego

WG6 – Performance Modeling, Metrics, and Specifications Guidelines and Questions

- As input to HECRTF charge (2c), provide information about the types of system design specifications needed to effectively meet various application domain requirements.
- Examine current state and value of performance modeling, metrics for HEC and recommend key extensions
- Analyze performance-based procurement specifications for HEC that lead to appropriately balanced systems.
- Recommend initiatives needed to overcome current limitations in this area.
- Example topics:
  - Metrics, time to solution, measurement and modeling methods, benchmarking, specification parameters, time to solution and relationship to fault-tolerance

Working Group Participants

- David Bailey
- Stan Ahalt
- Stephen Ashby
- Rupak Biswas
- Patrick Bohrer
  Carleton DeTar
- Jack Dongarra
- Ahmed Gameh
- Brent Gorda
- Adolfy Hoisie
- Sally McKee
- David Nelson
- Allan Snavely
- Carleton DeTar
- Jeffrey Vetter
- Theresa Windus
- Patrick Worley
- and others

Charter

- Establish objectives of future performance metrics and measurement techniques to characterize system value and productivity to users and institutions. Identify strategies for evaluation including benchmarking of existing and proposed systems in support of user applications. Determine parameters for specification of system attributes and properties.

Fundamental Metrics

Best single overriding metric: **time to solution**.

Time to solution includes:

- **Execution time.**
- Time spent in batch queues.
- System background interrupts and other overhead.
- Time lost due to scheduling inefficiencies, downtime.
- Programming time, debugging and tuning time.
- Pre-processing: grid generation, problem definition, etc.
- Post-processing: output data management, visualization, etc.

Related Factors

- Programming time and difficulty
  - Must be better understood (and reduced).
  - Identify key factors affecting development time.
  - Identify HPC relevant techniques from software engineering.
  - Closely connected to research in programming models and languages.
- System-level efficiency
  - Some metrics exist (i.e. ESP).
- Performance stability
- Grid generation, problem definition, etc.
  - For some applications, this step requires effort more than the computational step.
  - No good metrics at present time.

Current Best Practice for Procurements

- Characterize machines via micro-benchmarks and synthetic benchmarks run on available machines.
  - Numerous general specifications.
  - Results on some standard benchmarks.
  - Results on application benchmarks (different for each procurement).
- Identify and track applications of interest.
  - Use modeling to characterize performance.
  - Validate models on largest available system of that kind.
- Optimization problem–solving with constraints, including performance, dollars, floor space, power.
  - This step is not standardized, currently ad hoc.

This approach is inadequate to select systems 10x or more beyond systems in use at a given point in time.

Toward Performance-Based System Selection

- Procurements or other system selections should **not** be based on any single figure of merit.
- Can various agencies converge on a reference set of discipline-specific benchmark applications?
- On a set of micro-benchmarks?
- How can we better handle intellectual property and classified code issues in procurements?

- Accurate performance modeling holds the best promise for simplifying procurement benchmarking.

Performance Modeling

- Goals: A set of low-level basic system metrics, plus a solid methodology for accurately projecting the performance of a specific high-level application program on a specific high-end system.
- Challenges:
  - Current approaches require significant skill and expertise.
  - Current approaches require large amounts of run time.
  - Fast, nearly automatic, easy-to-use schemes are needed.
- Benefits:
  - Architecture research
  - Procurements
  - Vendors
  - Users

Potential Modeling Impact

- Influence architecture early in the design cycle
- Improve applications development
  - Use modeling in the entire lifecycle of an application, including algorithmic selection, code development, software engineering, deployment, tuning.
- Impact assessment
  - Project new science enabled by a proposed petaflop system.
- Research needed in:
  - Novel approaches to performance modeling: analytical, statistical, kernels and benchmarks, synthetic programs.
  - How to deal with exploding quantity of performance data on systems with 10,000+ CPUs.
  - Online reduction of trace data.

System Simulation

Salishan Conference, Apr. 2003: “Computational scientists have become quite expert in using high-end computers to model everything except the systems they run on.”

Research in the parallel discrete event simulation (PDES) field now makes it possible to:

- Develop a modular open-source system simulation facility, to be used by researchers and vendors.
  - Prime application: modeling very large-scale inter-processor networks.
- Need to work with vendors to resolve potential intellectual property issues.

Tools and Standards

- Characterized workloads from different agencies
  - Establishing common set of low-level micro-benchmarks predictive of performance.
  - In-depth characterization of applications incorporated in a common performance modeling framework.
  - Enables comparability of models and cooperative sharing of workload requirements.
- A standardized simulation framework for modeling and predicting performance of future machines.
- Diagnostic tools to reveal factors affecting performance on existing machines .
- Intelligent, visualization-based facilities to locate “hot spots” and other performance anomalies.

Performance Tuning

- Self-tuning library software: FFTW, Atlas, LAPACK.
- Near-term (1-5 yrs):
  - Extend to numerous other scientific libraries.
- Mid-term (5-10 yrs):
  - Develop prototype pre-processor tools that can extend this technology to ordinary user-written codes.
- Long-term (10-15 yrs):
  - Incorporate this technology into compilers.

Example from history–vectorization:

- Step 1: Completely manual, explicit vectorization
- Step 2: Semi-automatic vectorization, using directives
- Step 3: Generate both scalar and vector code, selected with run-time analysis