Performance Contracts: Monitoring and Resource Management

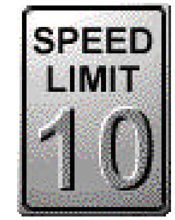
Toward Intelligent Software

Dan Reed
Department of Computer Science
National Center for Supercomputing Applications
University of Illinois

http://hipersoft.rice.edu/stc_site_visit/talks/Contracts.pdf

Intelligent Software: An Analogy

- 50 MPH is a legal stricture with no ambiguity
  - 51 MPH is a violation and you could be cited and fined
    - rarely are violators ticketed for such small violations
  - context determines actual behavior
    - city rush hour traffic rarely obeys speed limits
    - hazardous conditions change the effective speed limit

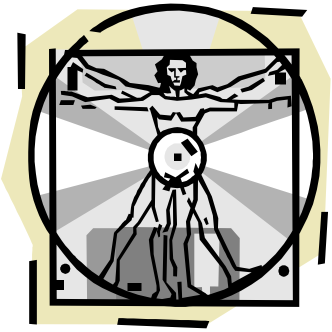


- What really happens
  - police use contextual discretion
    - “small” violations for “reasonable intervals” are tolerated
      - rush hour, weather, and special events
  - obeying the spirit of the law is usually the correct thing
    - perturbations about the limits are expected and accepted
  - if something happens, you want justice, not the law

- Intelligent, adaptive software is similar
  - application needs and available resources determine behavior
  - contracts must be flexible, with contextual discretion

Toward Intelligent Performance Toolkits

- Performance tools for computational grids
  - Grid environments are dynamic
  - applications and computational resources are also dynamic
    - must adapt to sustain predictable levels of performance
  - prerequisite of adaptation is recognition of changing conditions
- Approach
  - performance contract
    - specifies application and resource commitments
  - application and execution signature models
    - predict application and resource behavior
  - monitoring and forecasting infrastructure
    - detects when actual and predicted behaviors do not match
- Contract specification model options
  - measurement and forecasts, compiler, library, and/or user
  - historical, current, and predicted data



Sustaining Predictable Performance

- Detect if actual performance deviates from expected performance
  - prerequisites
    - prediction of expected performance
    - measurement of actual performance
- Identify the cause(s) of the deviation(s)
  - unexpected application behavior
  - poor prediction of expected performance on allocated resources
  - unanticipated load on one or more of the resources
    - in the extreme case, resource failure
- Provide information to help guide possible actions
  - migrate to new resources, continue on current resources, halt
  - switch to alternate algorithms or re-optimized code
  - lower precision of computations
- Archive collected information to improve future behavior
  - predictions, resource selection, and algorithms
  - application mixes and implicit interdependencies

Performance Contract Components

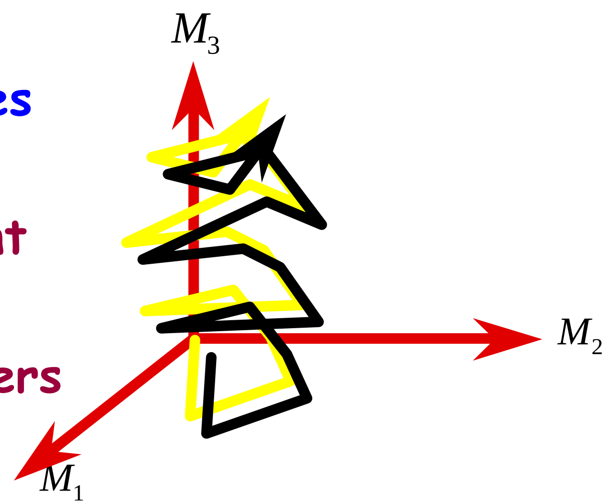
- Given
  - a set of resources (compute, network, I/O, …)
  - with certain capabilities (FLOP rate, latency, …)
  - for given application parameters (matrix size, image resolution, …)

  the application will
  - exhibit a specified, measurable, and desirable performance
    - sustain F FLOPS/second, render R frames/second, …

  as predicted by the model(s) (global composition of models)
- Performance contracts specify a convolution of
  - application intrinsic behavior and system resource responses
- Monitoring infrastructure verifies contract fulfillment
  - performance sensors inserted/activated where needed
    - real-time measurement and forecasting
      - application, systems, resources (NWS, …)
  - contract monitor detects when
    - actual and predicted behaviors do not match

Application and Execution Signatures

- **Application intrinsic metrics**
  - description of application demands on resources
  - sample metrics
    - bytes/message or FLOPS/source statement
  - values are independent of execution platform
    - but they may depend on problem parameters



- **Execution space metrics**
  - reflect application demands and resource response to those demands
  - express rates of progress
  - sample metrics
    - instructions/second or messages/second
  - values are dependent on execution platform
  - quantify actual performance and may include application interplay
- **Application and execution signatures**
  - trajectories of values through N-dimensional metric space

Example Performance Prediction Strategy

- Application signature model approach (very simplistic)
  - application signature defined by application code and parameters
  - application signature projected into execution metric space
    - scaling factors for each dimension (simplistic for many reasons)

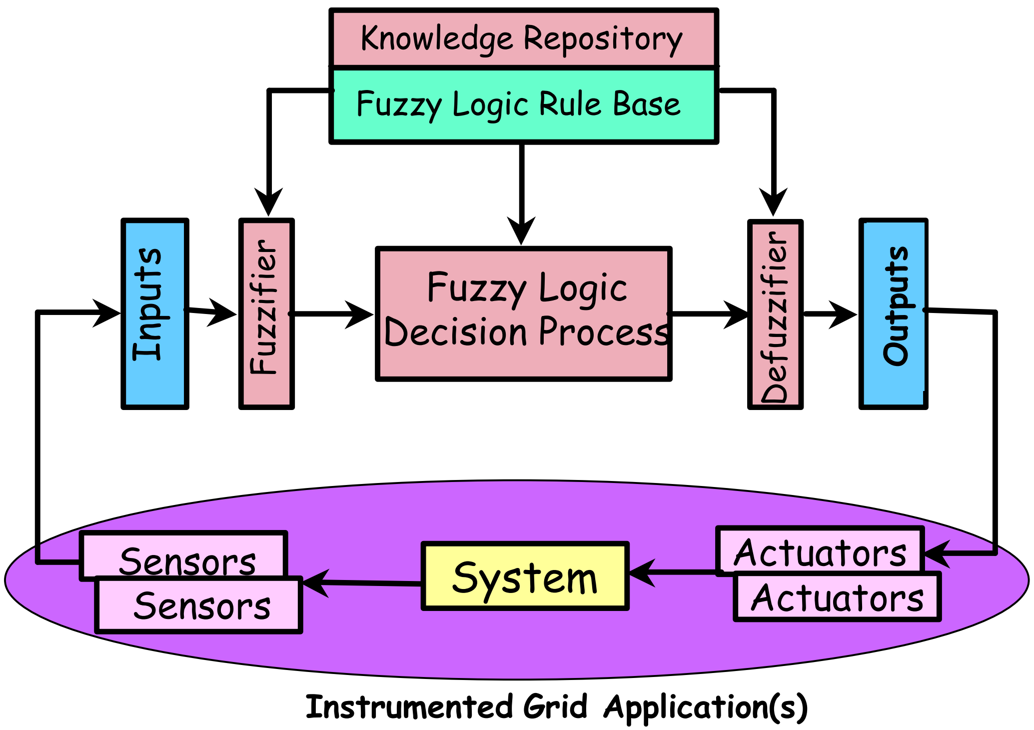
$$\frac{\text{statements}}{\text{FLOP}} \times \frac{\text{FLOPs}}{\text{second}} = \frac{\text{statements}}{\text{second}}$$

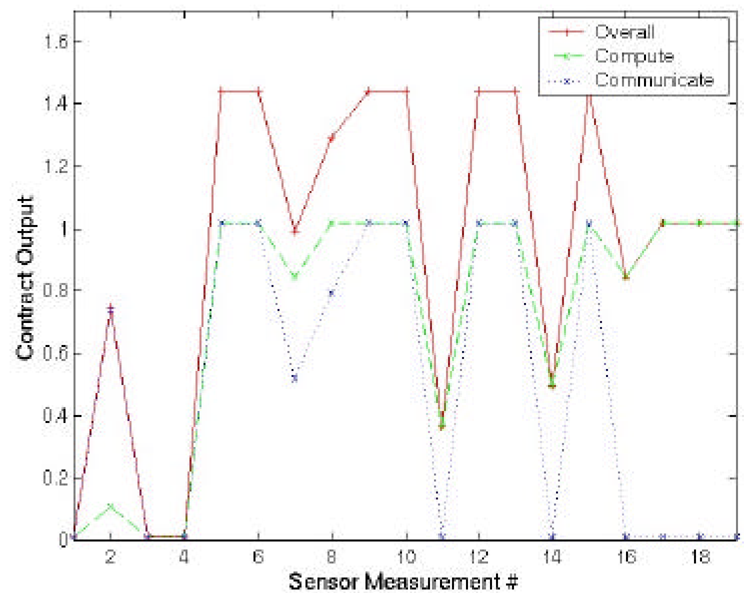
Application intrinsic	Projection factor	System specific

- Projection factors
  - correspond to capabilities of resources allocated to execution
  - express contract resources and capabilities
- Resulting execution signature
  - predicts application performance on given set of resources
  - expresses contract specified measurable performance

Example Contract Validation

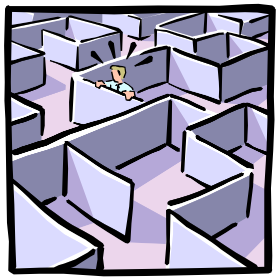
- ScaLAPACK PDGSEV execution
  - three separated Linux clusters
  - application intrinsic metrics
    - PAPI, MPI, and Autopilot
- Experiments
  - projections derived from baseline run
  - injected load on one node
    - induced perturbations elsewhere





Lessons Learned

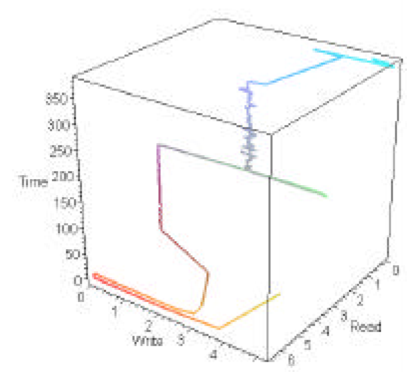
- Contracts provide a formalism for reasoning about behavior
  - spatial and temporal variability must be captured
  - even "simple" applications are surprisingly complex



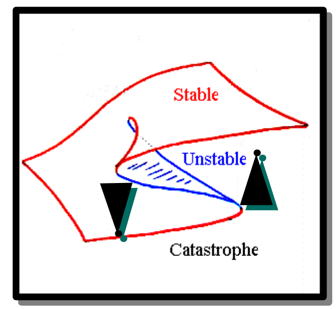
- Qualitative correctness is subtle and challenging
  - algorithmically describing acceptable behavior is challenging
  - violation severity, frequency, and sources must be specified
- Remediation has many levels and costs
- Separation of application and system specifications is critical
  - multivariate behavioral projections are needed
- Strong dependence on all software components
  - experiments require diverse software and research skills
- Testbeds really matter
  - controlled and large-scale for application validation and testing

Technical Challenges for the Future

- Signatures and projections
  - — multivariate projection and metric selection
  - — compact behavioral description
    - – polylines and feature extraction
  - — historical context from previous executions
  - — global temporal behavior and global evaluation
    - – multiple application and component interactions



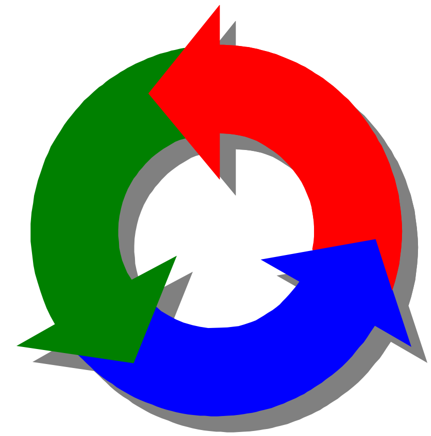
- Software infrastructure for distributed measurement
  - — correlation and extraction
  - — hierarchical contracts and management



- Contracts for application communities and resources
  - — data, networking, computation, visualization, …
- Grid economies, learning, and control systems
  - — learning techniques and generalization
  - — resource negotiation and validation
  - — grid dynamics and stability
  - — global efficiency and temporal evolution

Center Motivation and Needs

- Contract data sources
  - historical and predicted data on applications and systems
    - resource and application measurement expert engagement
  - deep compiler analysis and specifications
    - compiler expert engagement
    - application and library developer engagement
- Runtimes and environments
  - configurable object programs
    - adaptation and recompilation
  - schedulers and resource managers
    - infrastructure and policies for coordination
- Applications and testbeds
  - complex, multidisciplinary applications
    - engaged scientific community
  - realistic hardware/software testbeds
    - controlled environments for testing and experiments at scale



Interdependent components