Performance Modeling and Contracts

Ruth Aydt Dan Reed
Celso Mendes Fredrik Vraalsen

Pablo Research Group

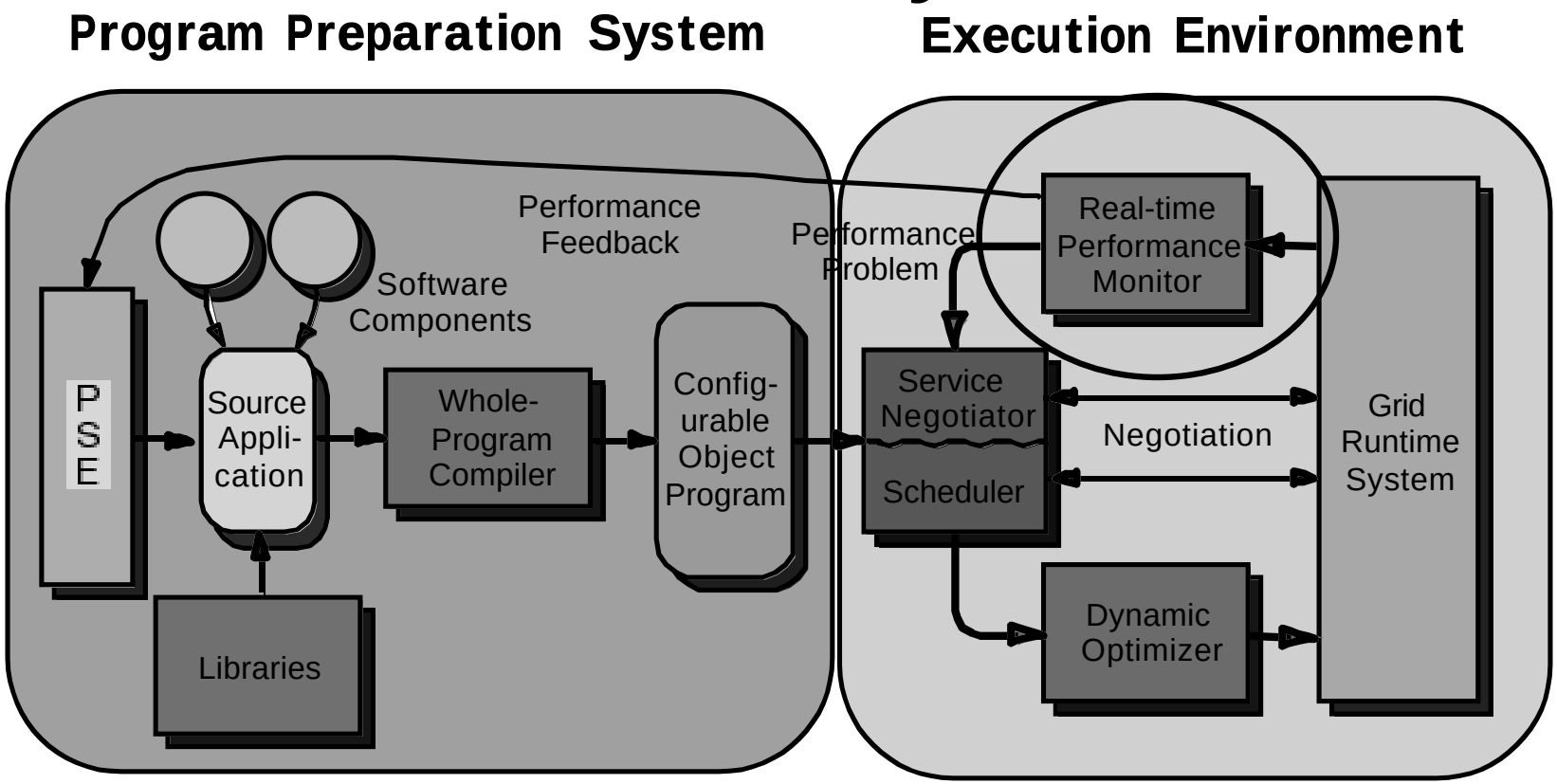
Department of Computer Science

University of Illinois

{aydt,reed,cmendes,vraalsen}@cs.uiuc.edu

http://www-pablo.cs.uiuc.edu

Making the Real-time Performance Monitor a Reality



Requirements

- Performance models that predict application performance on a given set of Grid resources
- Tools and interfaces that allow us to
  - Monitor application execution in real-time
  - Detect when observed performance does not match predicted performance
  - Identify cause of problem
  - Provide feedback to guide dynamic reconfiguration

Sources of Performance Models

- Developer knowledge of application or library behavior *ScaLAPACK – Jack's team*
  - Considerable detail possible
- Compile time analysis of code *Rice team - Keith*
  - Use compiler understanding of code behavior to build performance predictions
- Historical data from previous runs or observed behavior of current execution 'so far' *Pablo group*
  - "Learn" from past experience

Contract Specification

- "Boilerplate" for specifying performance model inputs and outputs; enumerates what all parties are committing to provide
- Given
  - a set of *resources* (compute, network, I/O, ...) *Fine Grid*
  - with certain *capabilities* (flop rate, latency, ...) *NWS*
  - for particular *problem parameters* (matrix size, image resolution, ...) *part of job submission*

the application will

  - achieve a *specified, measurable*

Real-time Performance Monitor

- Decide if the contract has been violated
- Strictly speaking, the contract is *violated* if any of the *resource, capability, problem parameter* or *performance specifications* are not met during the execution
- In practice, *tolerate* a level of contract violation
  - *specifications will have inaccuracies*
- The contract *violation policy* should consider the
  - *severity*
  - *persistence*
  - *cumulative effect*

of the breach of contract in determining when

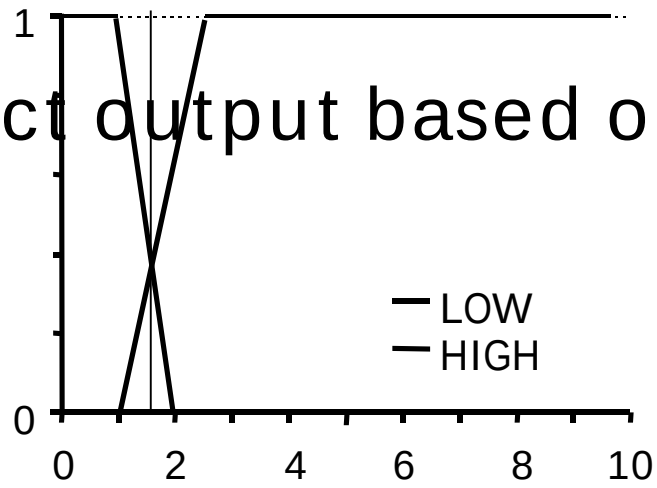
“Tunable” Tolerance

Approach:

- Use *Autopilot* decision procedures that are based on fuzzy logic to deal with uncertainty
- These support intuitive reasoning about degrees of violation
  - “if FLOP rate has been low for a long time the contract is violated”
  - what constitutes low, long, and violated can be adjusted to express different levels of uncertainty and tolerance
  - can also set threshold on ‘how bad it must be violated’ before it is actually reported
  - many knobs to turn!
- Violation transitions are smooth rather than discrete as they are with decision tables

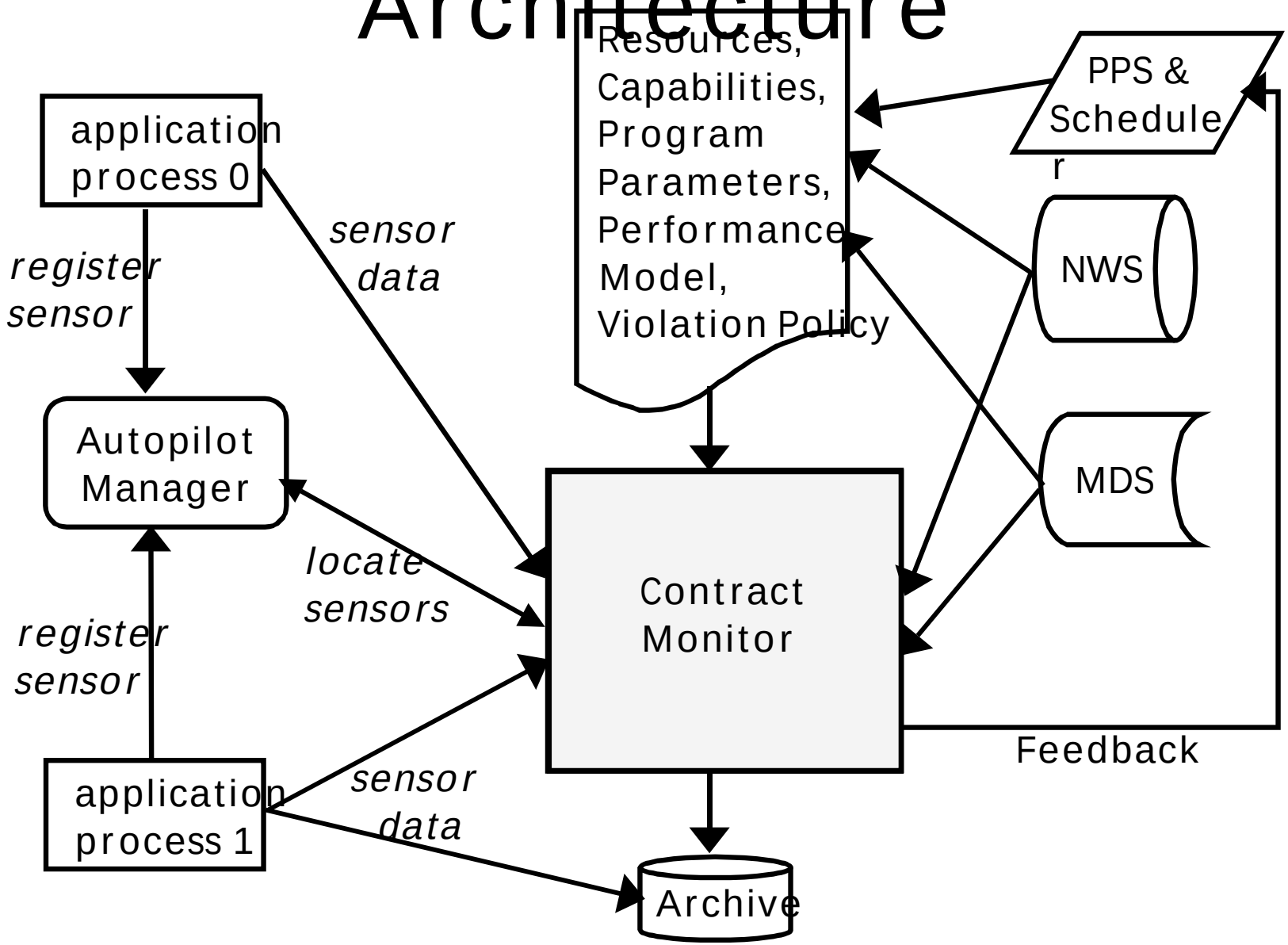
ScaLAPACK Developer Model

- Model predicts duration for each iteration
- An Autopilot Sensor inserted in application reports iteration number and actual duration
- Contract Monitor computes ratio of actual to predicted time; ratio passed to decision procedure
- Fuzzy rules specify contract output based on ratio



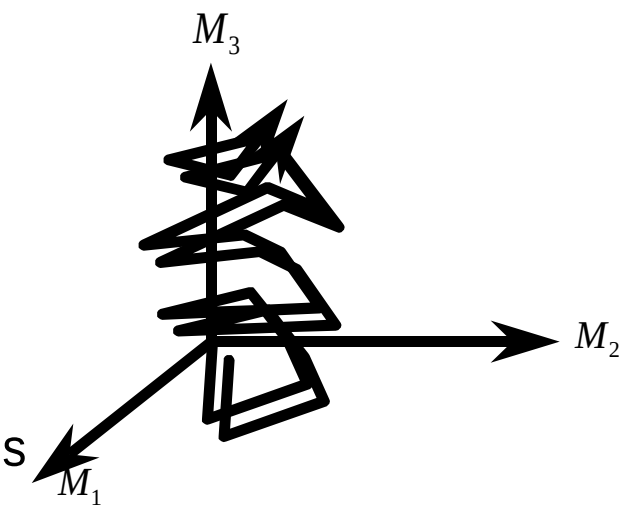

```
var timeRatio ( 0, 10 ) {
  set trapez LOW  ( 0,  1, 0, 1 );
  set trapez HIGH ( 2, 10, 1, 0 ); };

var contract ( -1, 2 ) {
  set triangle OK         ( 1, 1, 1 );
  set triangle VIOLATED ( 0, 1, 1 ); };
```

Contract Monitoring Architecture



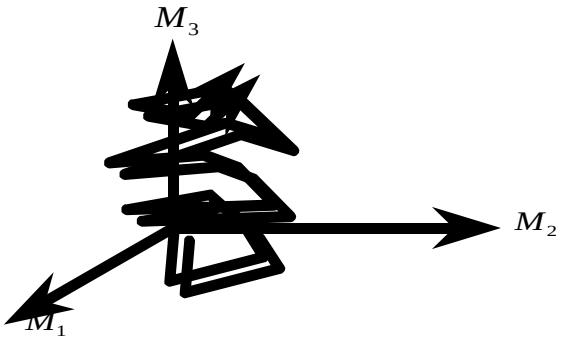
Application Signature Model

- Application Intrinsic Metrics
  - description of application demands on resources
  - sample metrics
    - » FLOPS/statement, I/O bytes/statement, bytes/message
  - values are independent of execution platform
  - values may depend on problem parameters
- Application Signature
  - trajectory of values through N-dimensional metric space
  - one trajectory per task
    - » correlated for data parallel codes



System Space Signature

- System Space Metrics
  - description of resource response to application demands
  - sample metrics
    - » FLOPS/second, I/O bytes/second, message bytes/second
  - values are dependent on execution platform
  - quantify actual performance
- System Space Signature
  - trajectory of values through N-dimensional metric space
  - will vary across application executions, even on the same resources



Performance Prediction Strategy

- Given
  - application intrinsic behavior
  - resource capability information
  - *project* application signature into system space signature, in effect *predicting performance*
- Many possible projection strategies
  - single figure of merit (scaling in each dimension)
    - » peak MFLOPS, bandwidth, I/O rate
    - » benchmark suite measurements
    - » previous application executions (learning)

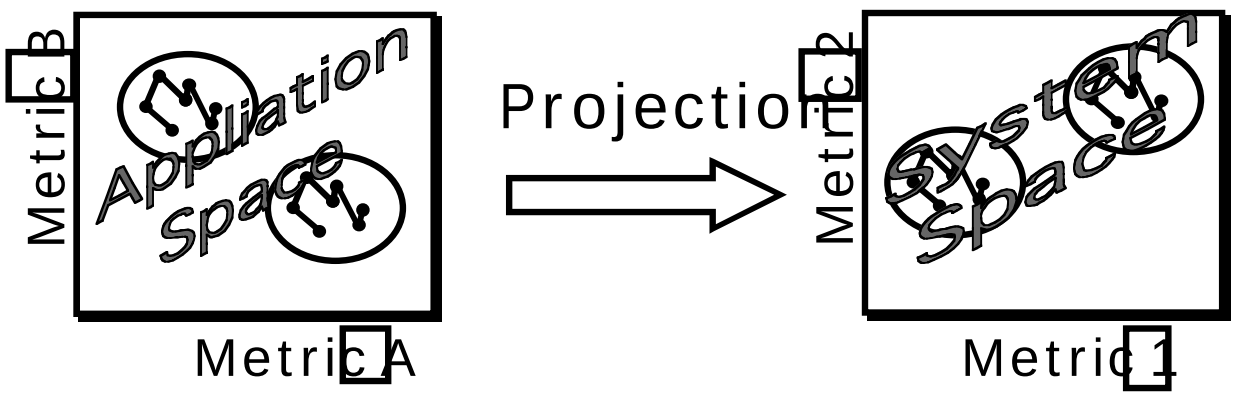
Single Figure of Merit Projection

A $\frac{\text{Instructions}}{\text{FLOPS}} \times \frac{\text{FLOPS}}{\text{Second}} = \frac{\text{Instructions}}{\text{Second}}$ 1

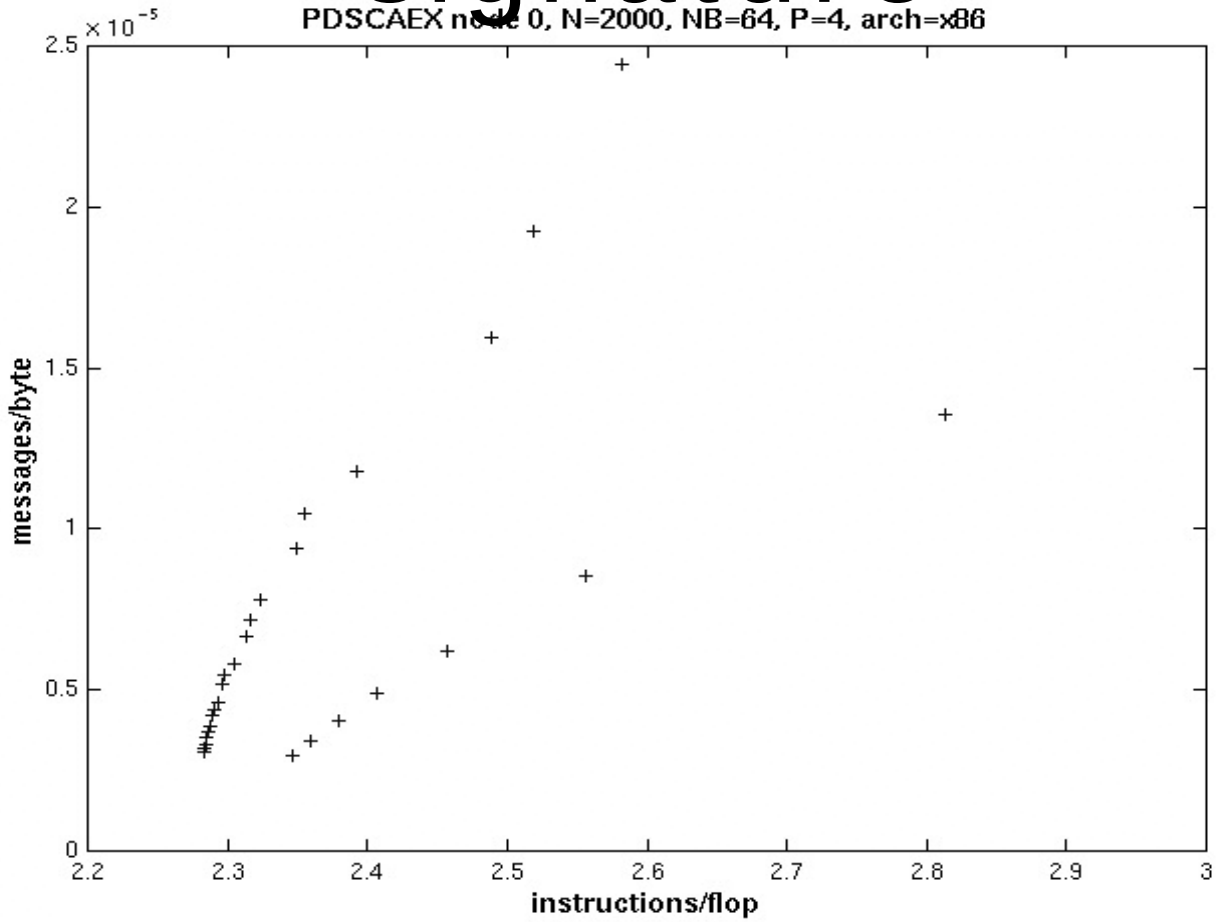
Application Intrinsic ↑ | Projection Factor ↑ | System Specific ↑

Application Intrinsic ↓ | Projection Factor ↓ | System Specific ↓

B $\frac{\text{Messages}}{\text{Byte}} \times \frac{\text{Bytes}}{\text{Second}} = \frac{\text{Messages}}{\text{Second}}$ 2

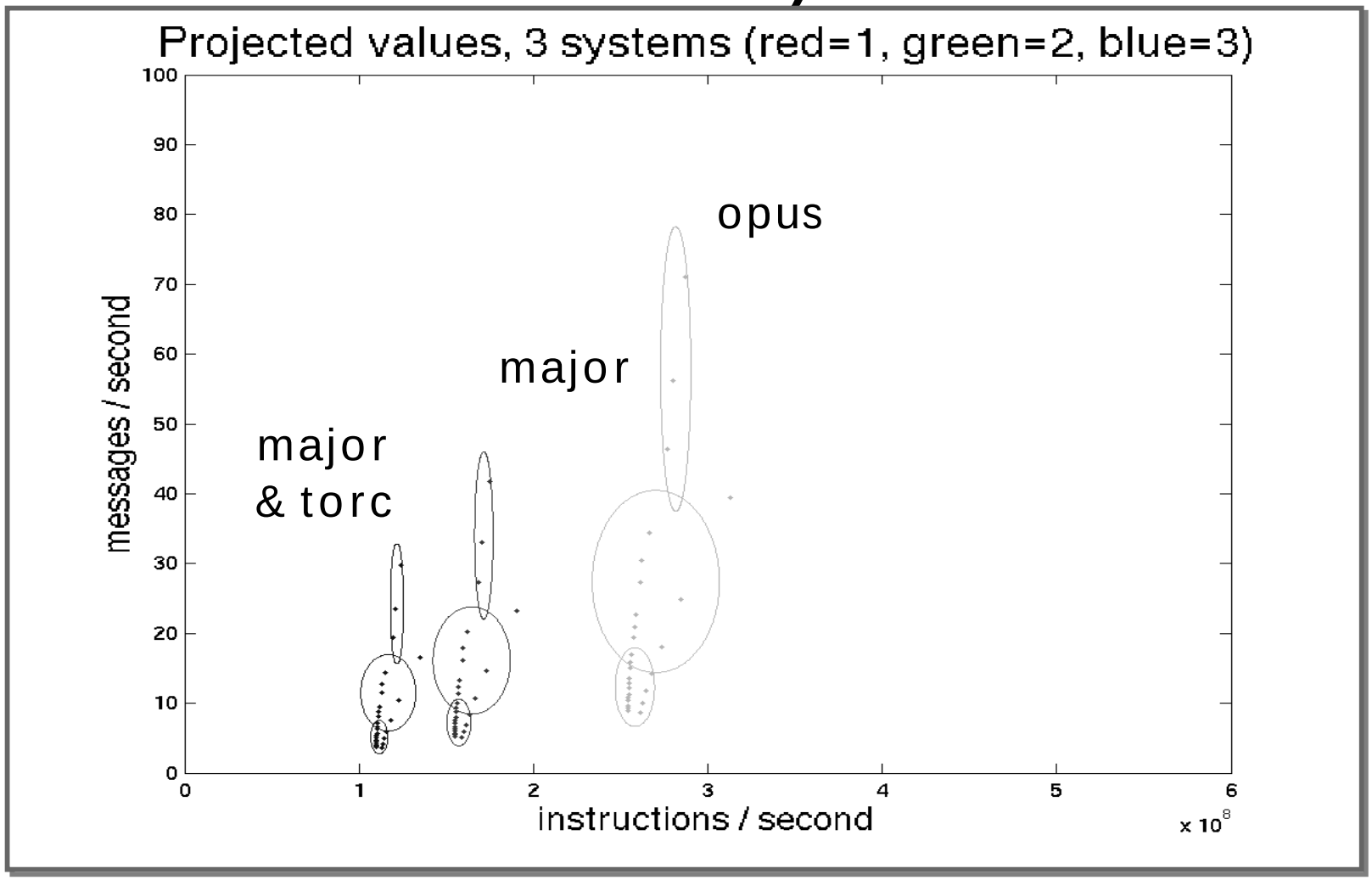


ScaLAPACK Application Signature



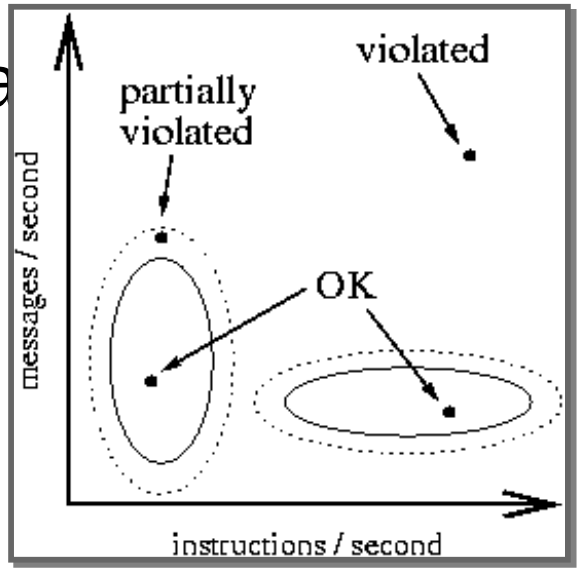
- System independent *modulo instructions versus statements*
- Trajectory reflects change in application demands over course

Projected Behavior (3 resource sets)



Contract Implementation

- Cluster the (projected) system space points
  - centroids define nominal predicted behavior
  - radii define ‘tolerable range’ of values
- Compare actual performance to cluster predictions
- If ‘far’ from cluster, report viola



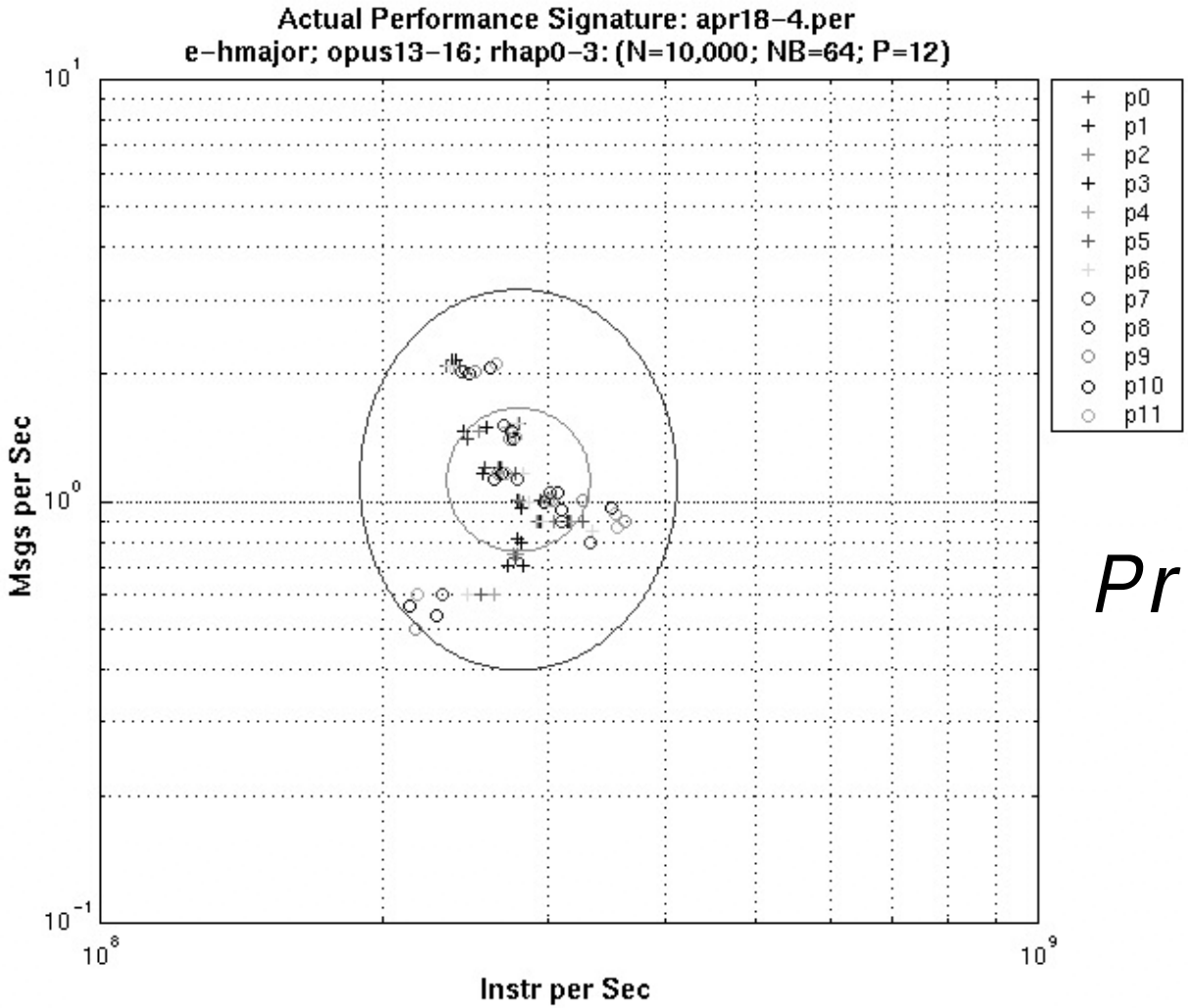

```
var distanceFromCentroid ( 0, 100 ) {
   set trapez SHORT  ( 0,   .3,  0,  .3 );
   set trapez LONG ( .6,  100,  .3,  0 ); };

var contract ( -1, 2 ) {
   set triangle OK          ( 1, 1, 1 );
   set triangle VIOLATED ( 0, 1, 1 ); };

if ( distanceFromCentroid == SHORT ) { contract = OK; }
if ( distanceFromCentroid == LONG) { contract = VIOLATED; }
```

Experimental Verification of Approach

ScaLAPACK Periodic Application Signature Model

- Application-Intrinsic metrics captured every 60 seconds using PAPI, MPI wrappers, Autopilot Sensors
- Projections based on historical data
- Run on 3 clusters at UIUC; used 4 machines from each cluster; machine speeds vary across clusters
- Introduced CPU load on one machine
- Contract Monitor detects violation

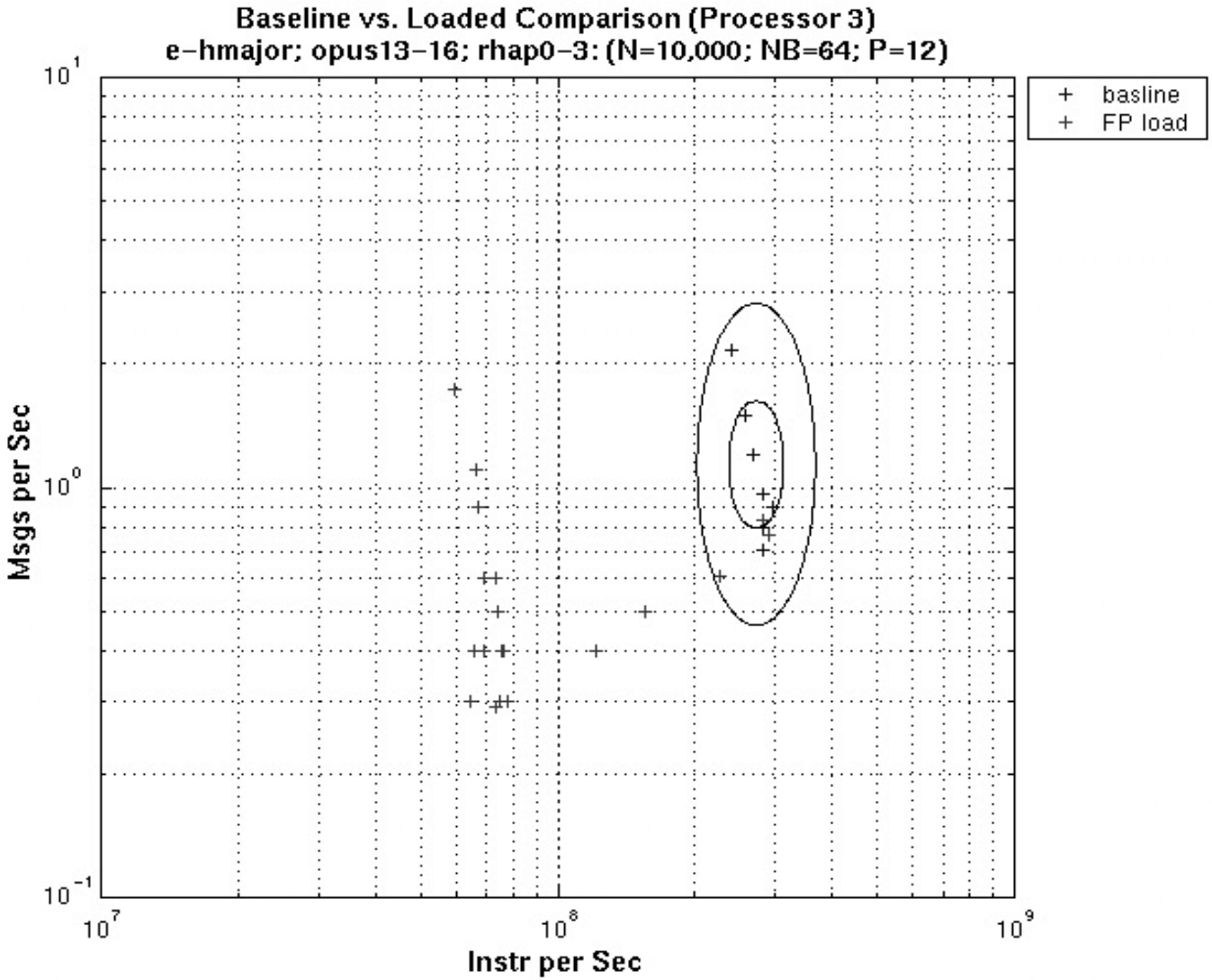
Projected & Actual Comparison:



Promising!

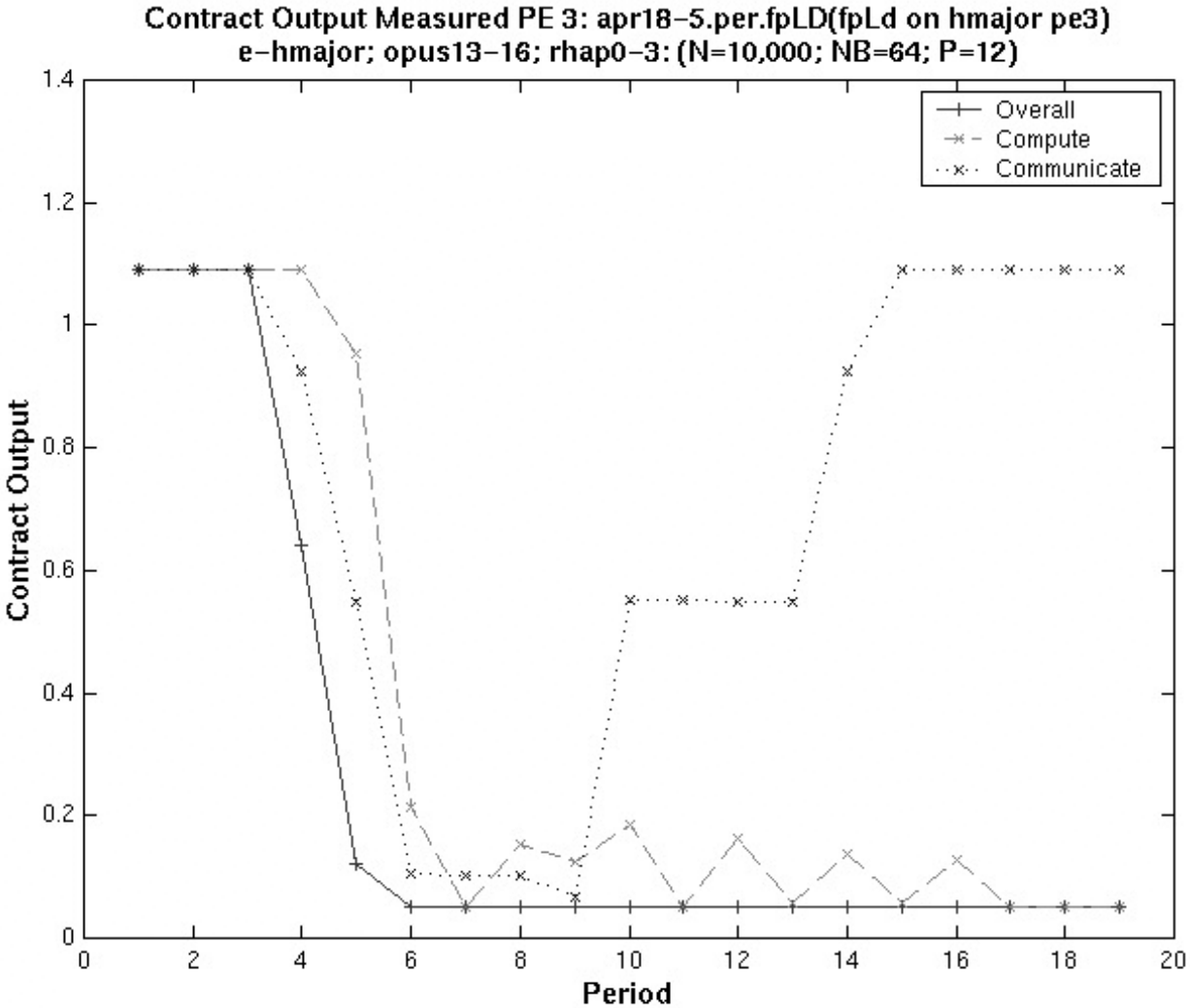
- Radii based on standard deviation of each projection factor
  Green – 2X std deviation; Red: 4X std deviation

Load on P3: Predicted & Measured



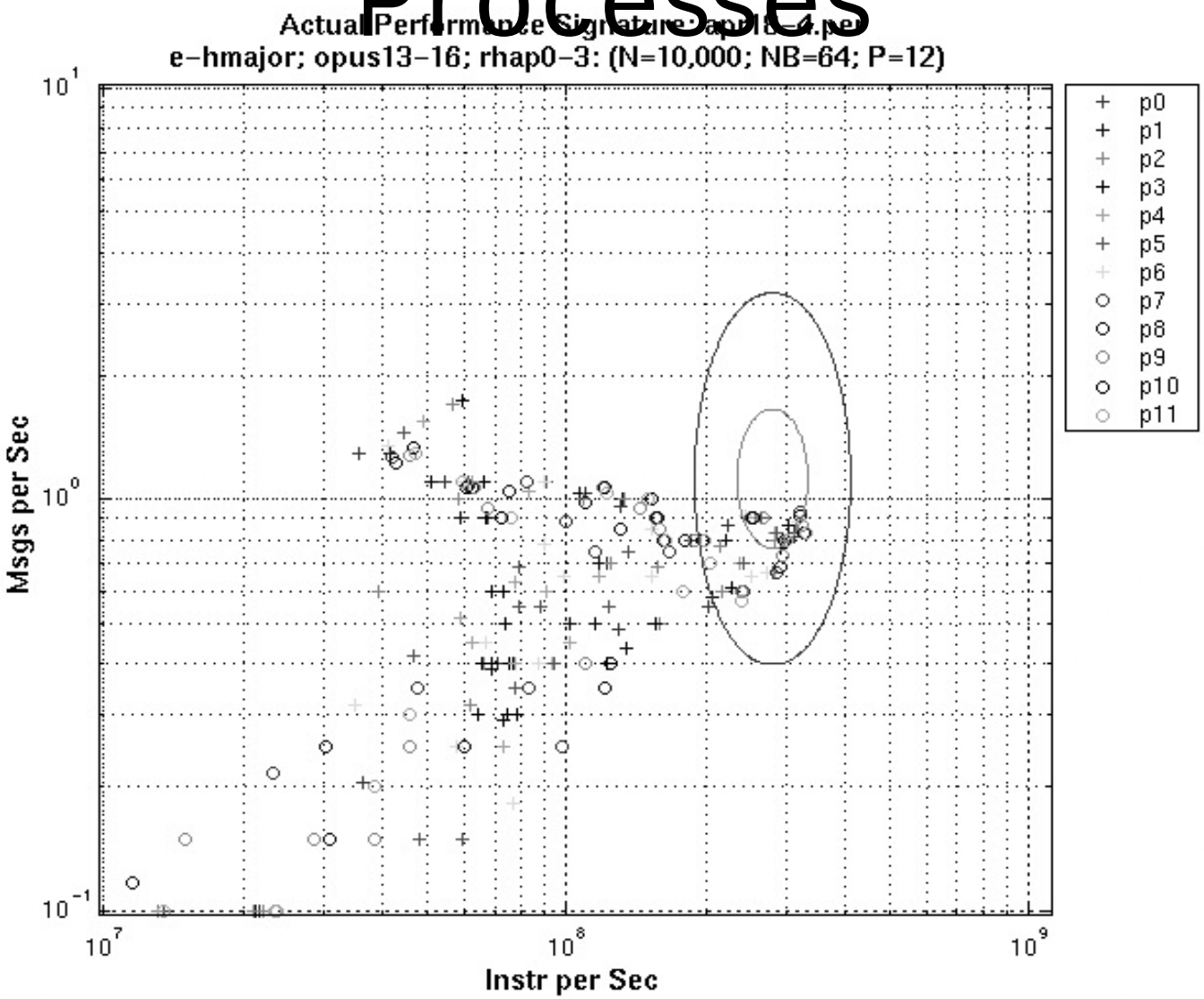
- Shift down and to the left (metrics not independent for ScaLAPACK)

P3 Contract Monitor Output



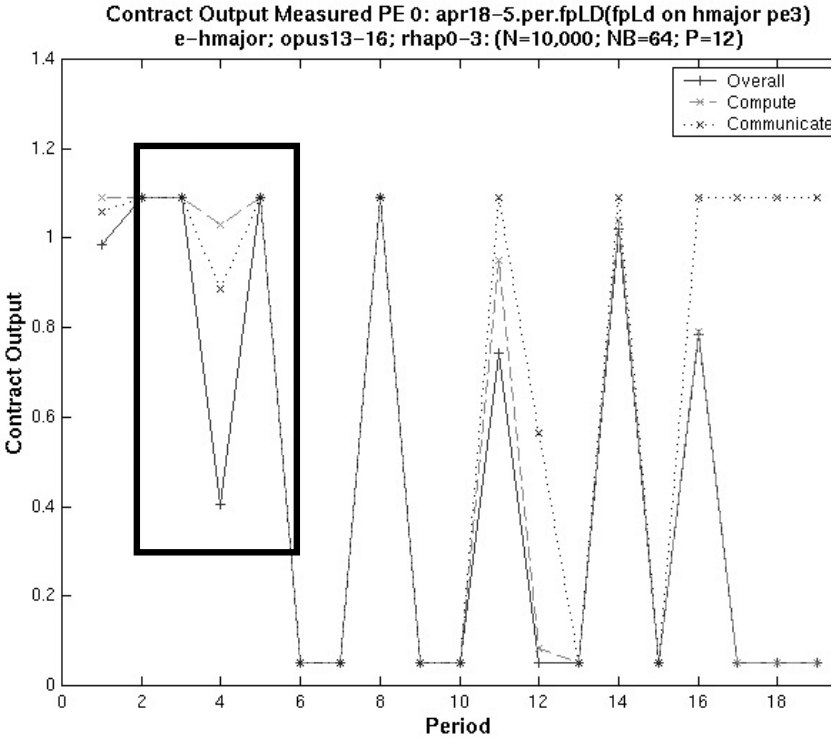
- Violations in System Space when load introduced (~3min)
- Also looking at compute and communicate

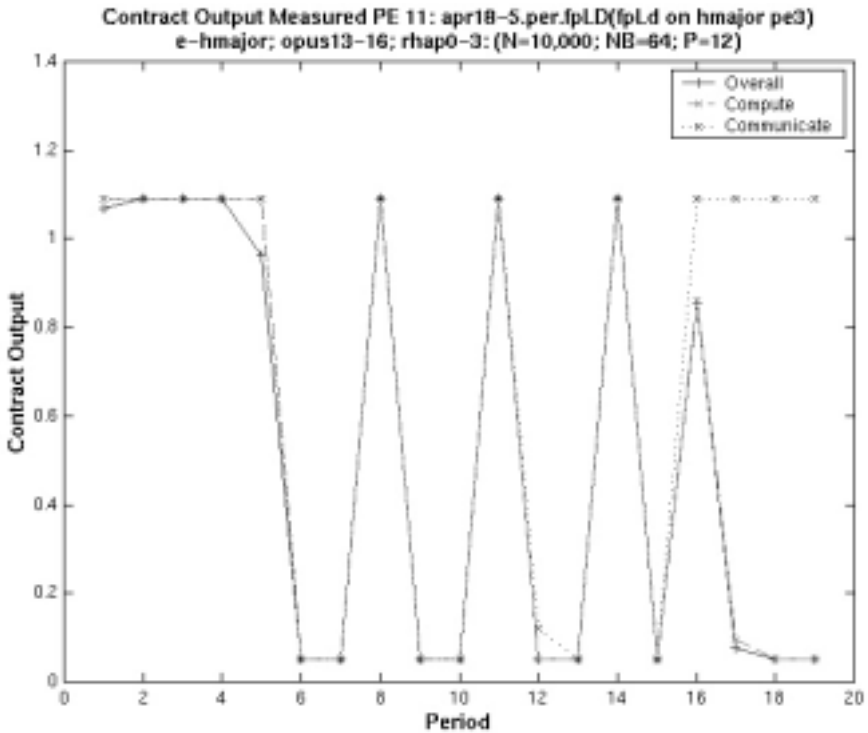
Load on P3: Impact on All Processes



- All shift to the left and down – *is detecting culprit hopeless?*

Contract Results P0, P11



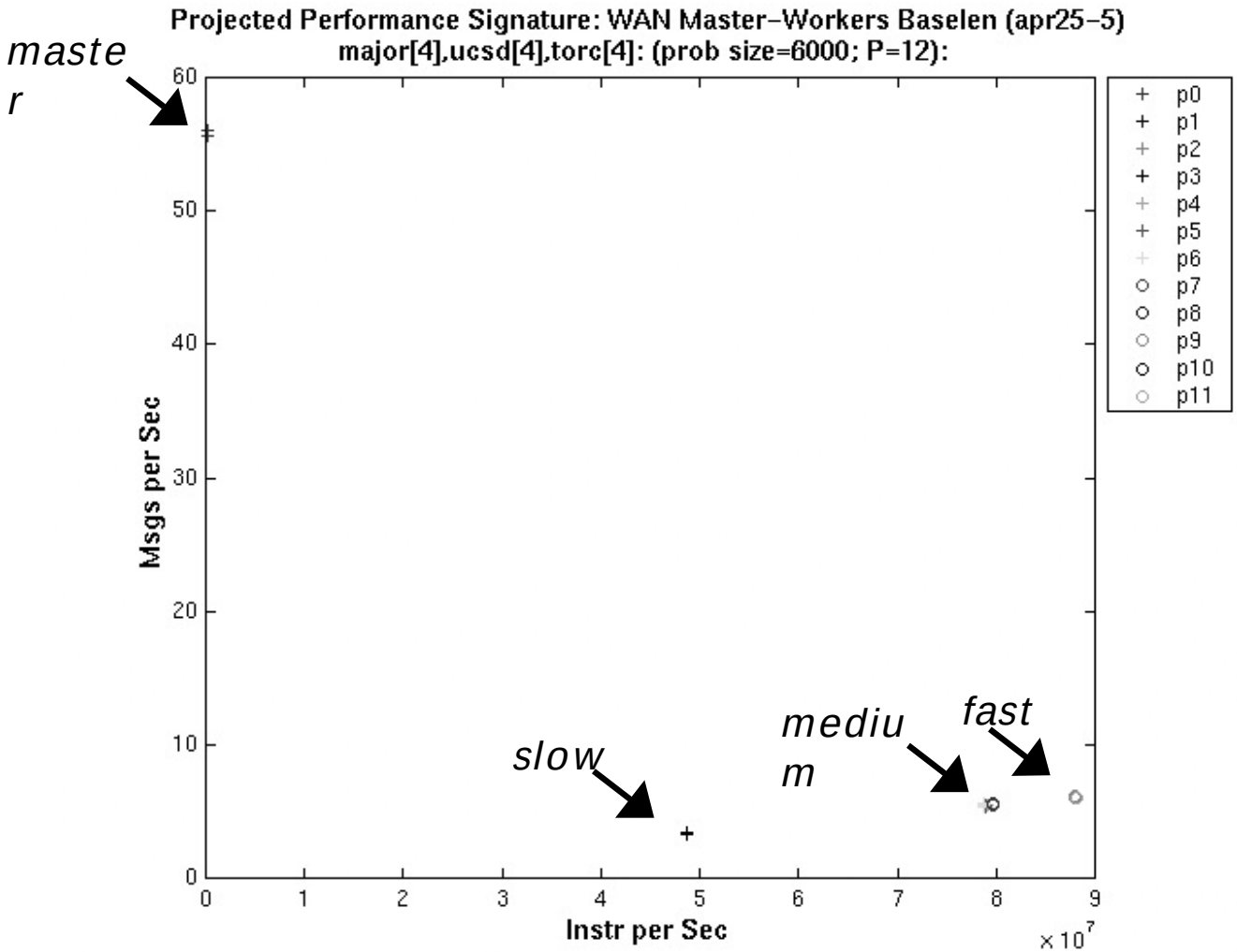


- *Perhaps there is hope!*
- Contract output for other processes not consistently violated
- Side Note: individual components "combine" for overall violation

Master/Worker Experiments

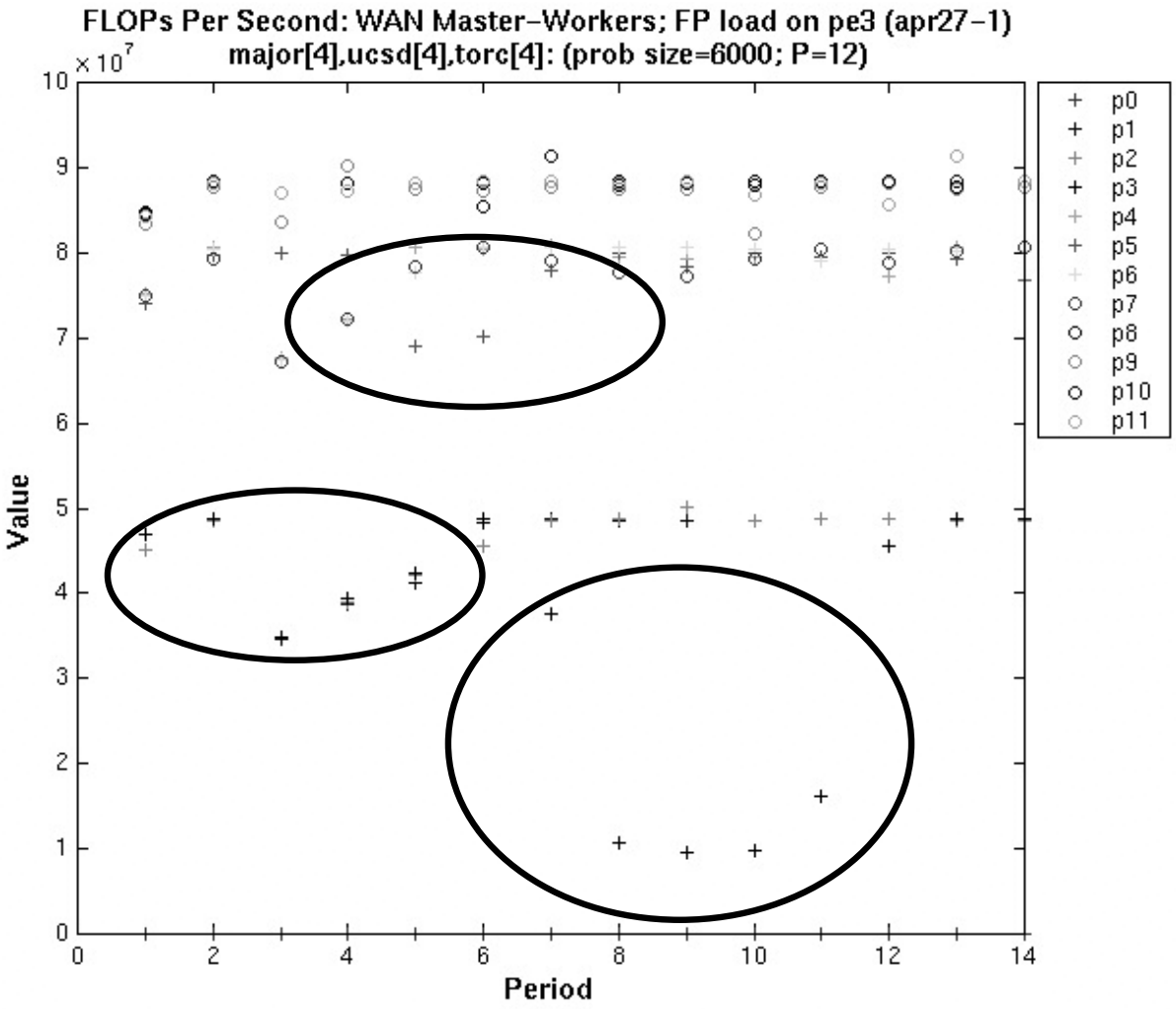
- Synthetic program
  - Master (p0) hands out fixed-size jobs via message to workers
  - Workers compute independently; report 'solution' back to master
  - Master may be bottleneck
  - Two classes of behavior among processes
- Results shown
  - Application-Intrinsic metrics captured every 30 seconds
  - Projections based on historical data from baseline execution
  - Execute in WAN (UIUC, UCSD, UTK)
  - Load introduced and then removed
  - Contract monitor detects violation
- MicroGrid will allow us to more easily conduct

Projected Performance



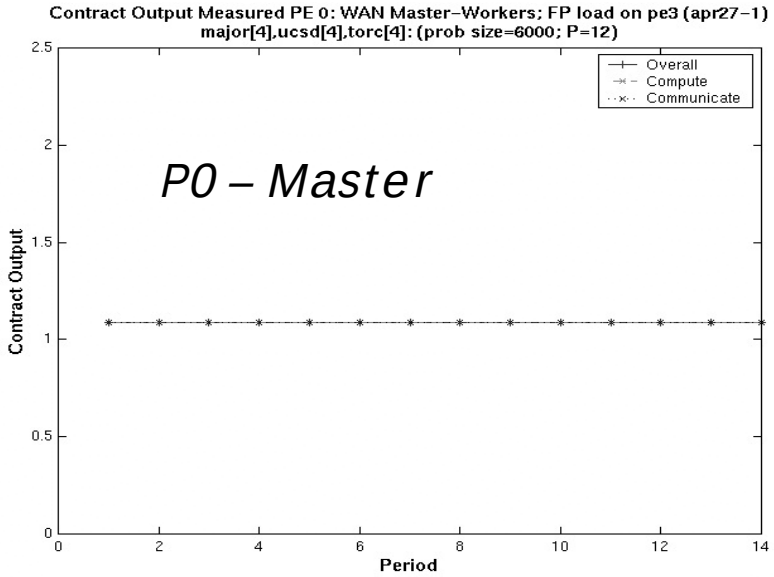
- Different Master and Worker behavior reflected
- Different Worker processor speeds reflected

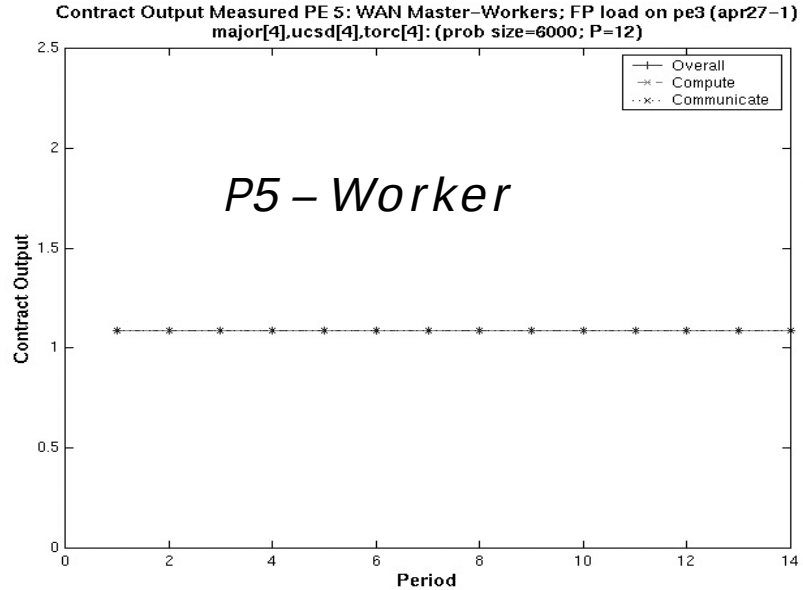
Measured FLOP Rate over Time

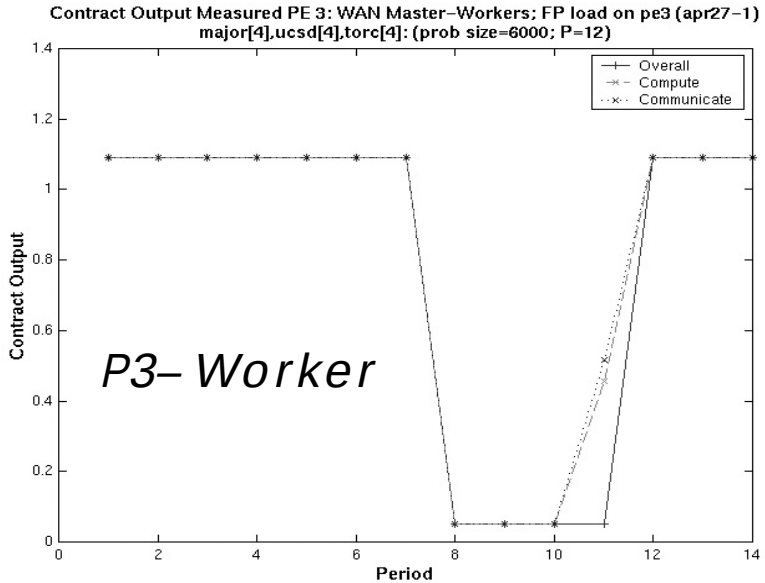


- Contract detects severe dip in P3 FLOP Rate
- Ignores small drop in P5 FLOP Rate and earlier drop in P3

Contract Output; load on P3







Where we are; where we're going

- Performance models
  - application signature models look promising
    - » Paper submitted to HPDC
    - » Explore periodic reclustering to capture temporal behavior evolution
    - » Determine if common set of metrics capture important characteristics of wide range of application types
  - use compiler insights to develop better contracts
- Contract monitoring infrastructure in place
  - lessons learned reflected in work of Global Grid Forum Performance WG
  - supports multi-level contract monitors as first step toward identifying per-process and overall violations
  - experiment with different violation boundaries
  - identify cause of violations; preliminary results reasonably good
- Research Topics
  - Development of methods to automatically select & tune fuzzy logic rulebases for different sets of resources; (Mario Medina)
  - Automatic detection of phase changes to trigger reclustering

Monitoring Progress: Are we meeting our Commitments?

Year 1:

- creation of initial performance models completed
- gain insight into performance of existing algorithms and libraries many insights from ScaLAPACK that are guiding what is possible to predict/detect overall
- specify interfaces for defining and validating performance contracts initial interfaces defined; continue to refine as we learn more about what is required to support wider range of contracts
- specify form and semantics of performance reporting mechanisms complete from application to contract monitor; feedback from monitor to PSE and Scheduler not done.

Year 2:

- real-time, wide-area performance measurement tools completed
- sophisticated application performance models that relate