



Performance Monitoring and Modeling to Drive Reactivity

Dan Reed

Department of Computer Science

University of Illinois

reed@cs.uiuc.edu

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



Major Contributors

- Ruth Aydt
 - software infrastructure
- Celso Mendes
 - clustering
- Fredrik Vraalsen
 - contracts
- Ying Zhang
 - measurement



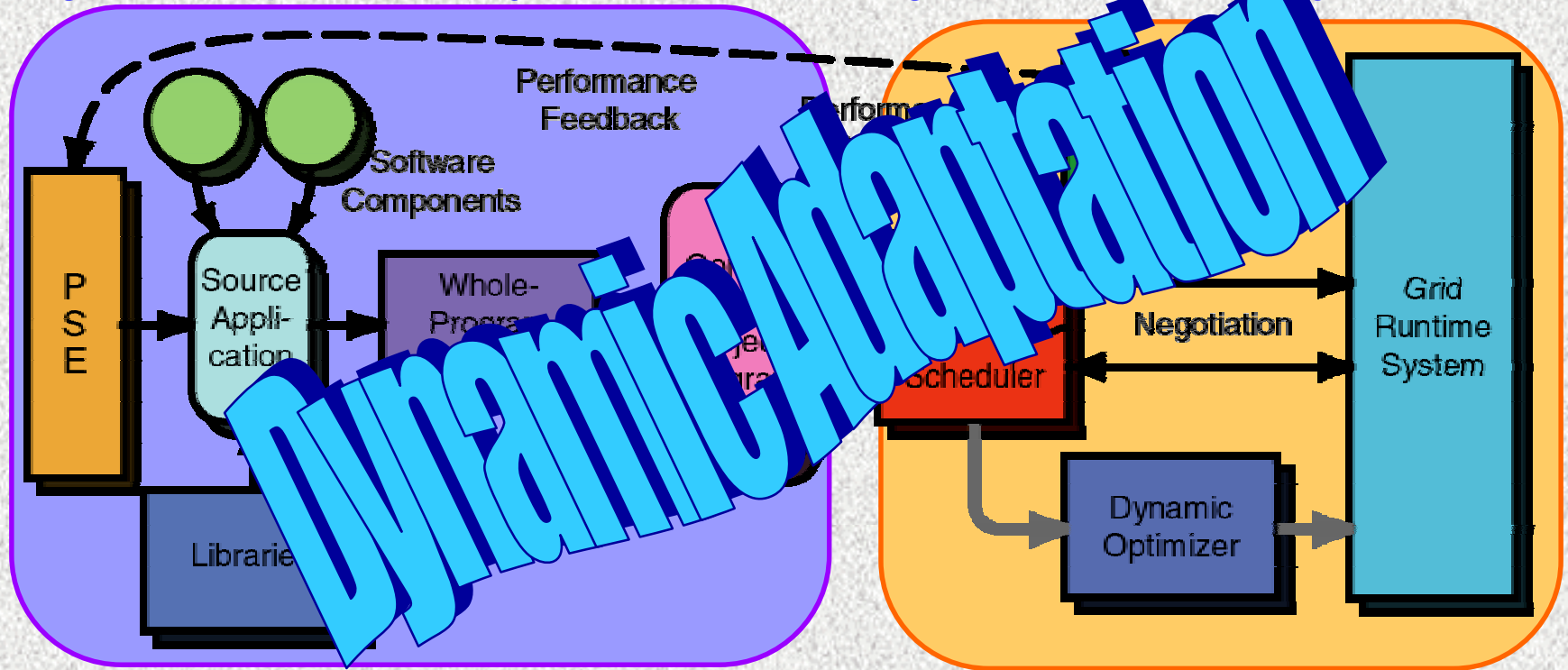
Presentation Outline

- Mini-tutorial
 - fuzzy logic and adaptive control
- Application signatures
 - execution context independent specification
 - clustering and projection pursuit
- Contract specification
 - clusters and fuzzy logic rules
- Performance experiments



The Grid "Big Picture"

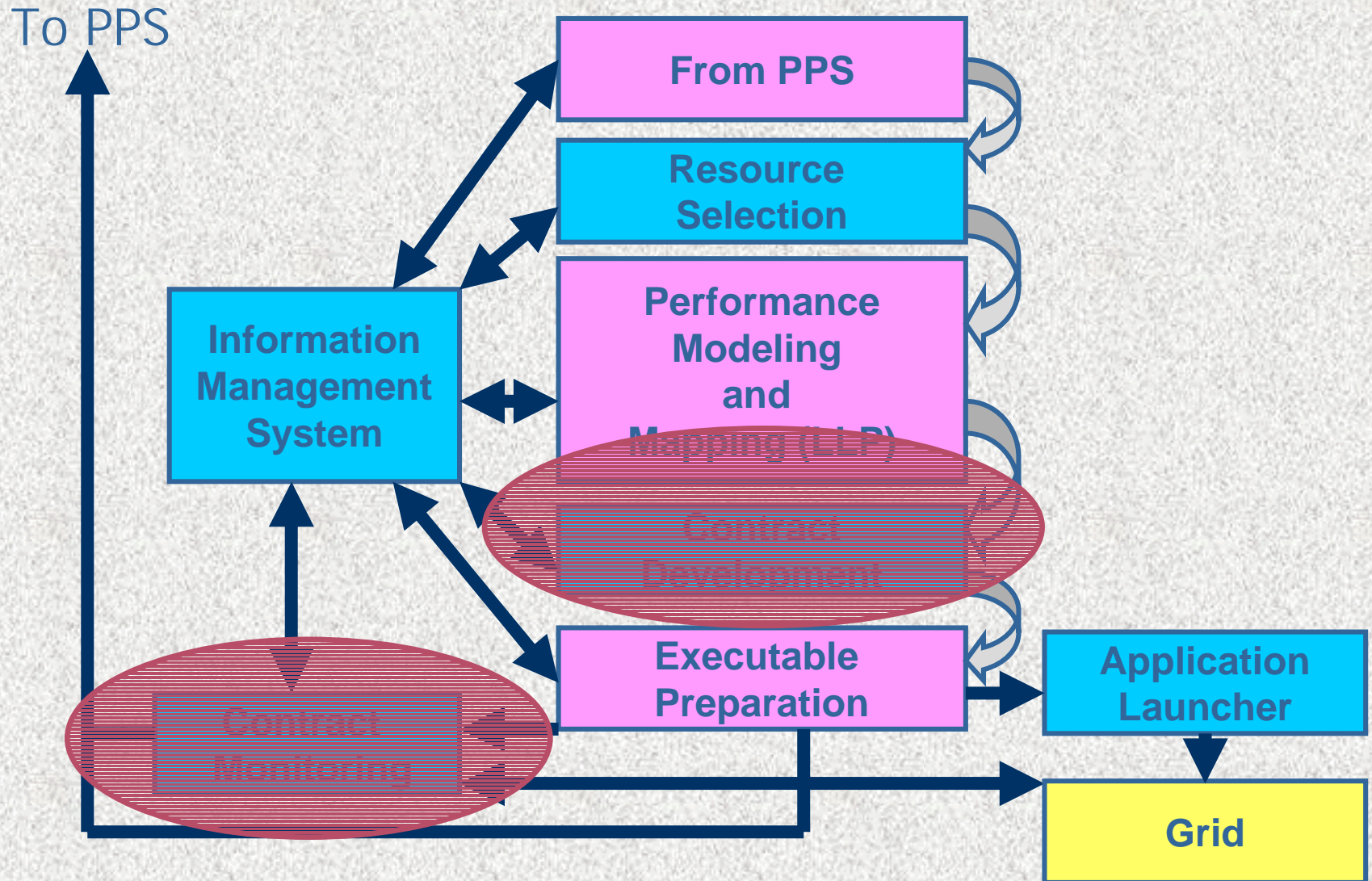
Program Preparation System (PPS) Program Execution System (PES)



Participants: Ruth Aydt, Andrew Chien, Fran Berman, Jack Dongarra, Ian Foster, Dennis Gannon, Ken Kennedy, Carl Kesselman, Lennart Johnsson, and Dan Reed



Program Execution System





Automatic Tuning Requirements

- End-to-end, real-time data capture
 - multiple levels
 - » hardware, system software, libraries, applications
 - multiple granularities
 - » microseconds to hours
 - multiple sites
 - » geographically distributed computations
- Intelligent data analysis
 - qualitative feature extraction
- Dynamic policy selection
 - interactive and automatic



Performance Contracts

- At the heart of the GrADS Model
 - mechanisms for managing mapping and execution
- What are they?
 - mappings from resources to performance
 - mechanisms for determining
 - » when to interrupt and reschedule
- Abstract definition
 - random variable: $r(A, I, C, t_0)$ with a PDF
 - » $A = \text{app}$, $I = \text{input}$, $C = \text{configuration}$, $t_0 = \text{time of initiation}$
 - » important statistics: lower and upper bounds (95% confidence)
 - issue:
 - » Is r a derivative at t_0 ? (Wolski)



Conceptual Contract Methodology

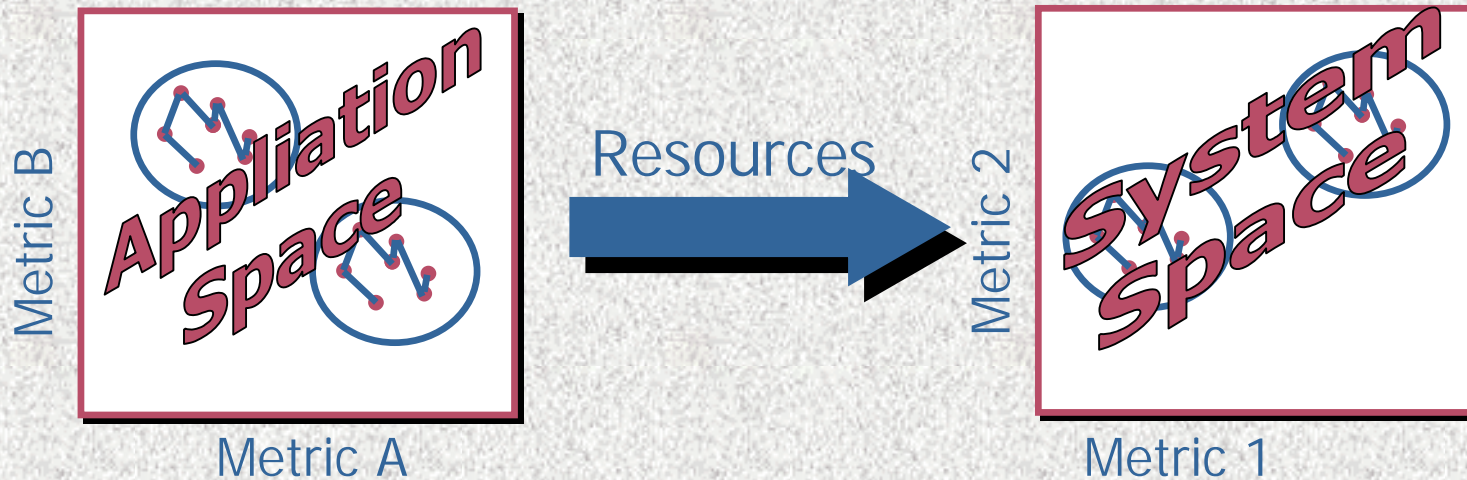
- Combine
 - application characteristics
 - » developer
 - » compile-time analysis
 - » previous application executions
 - resource specifications
 - » Network Weather Service (NWS)
 - » benchmarks, and other data
- Project expected performance
 - using characteristics and specifications (models)
- Measure run-time performance and test contracts
 - using application instrumentation
 - » detect violations (application, resource, and/or model)
 - » identify causes





Performance Contracts

- Given
 - compact application signature
 - clustering of application intrinsic metrics
 - resource behavioral estimates
- Project application signature
- Validate projection against measured behavior

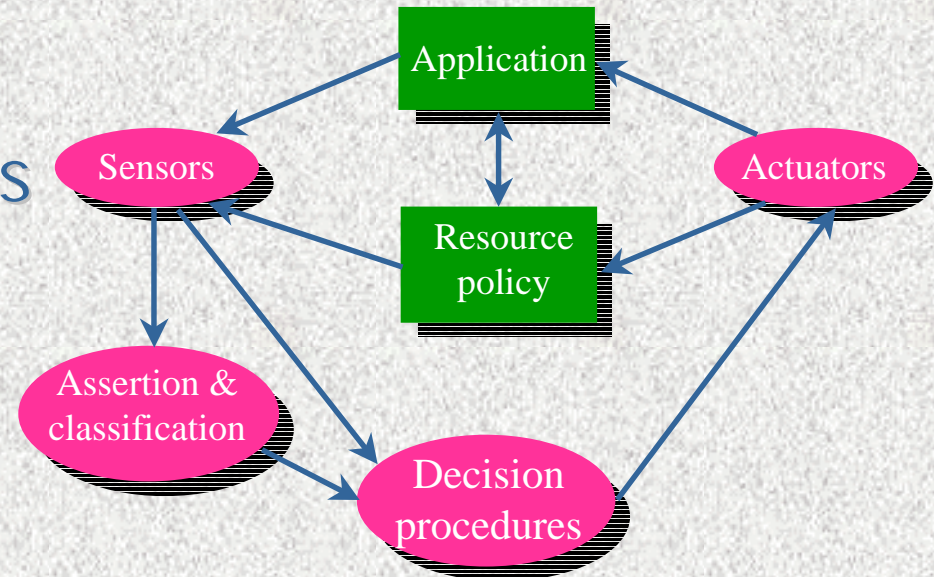




Autopilot Adaptive Control

■ Rationale

- adaptive applications
- dynamic demands

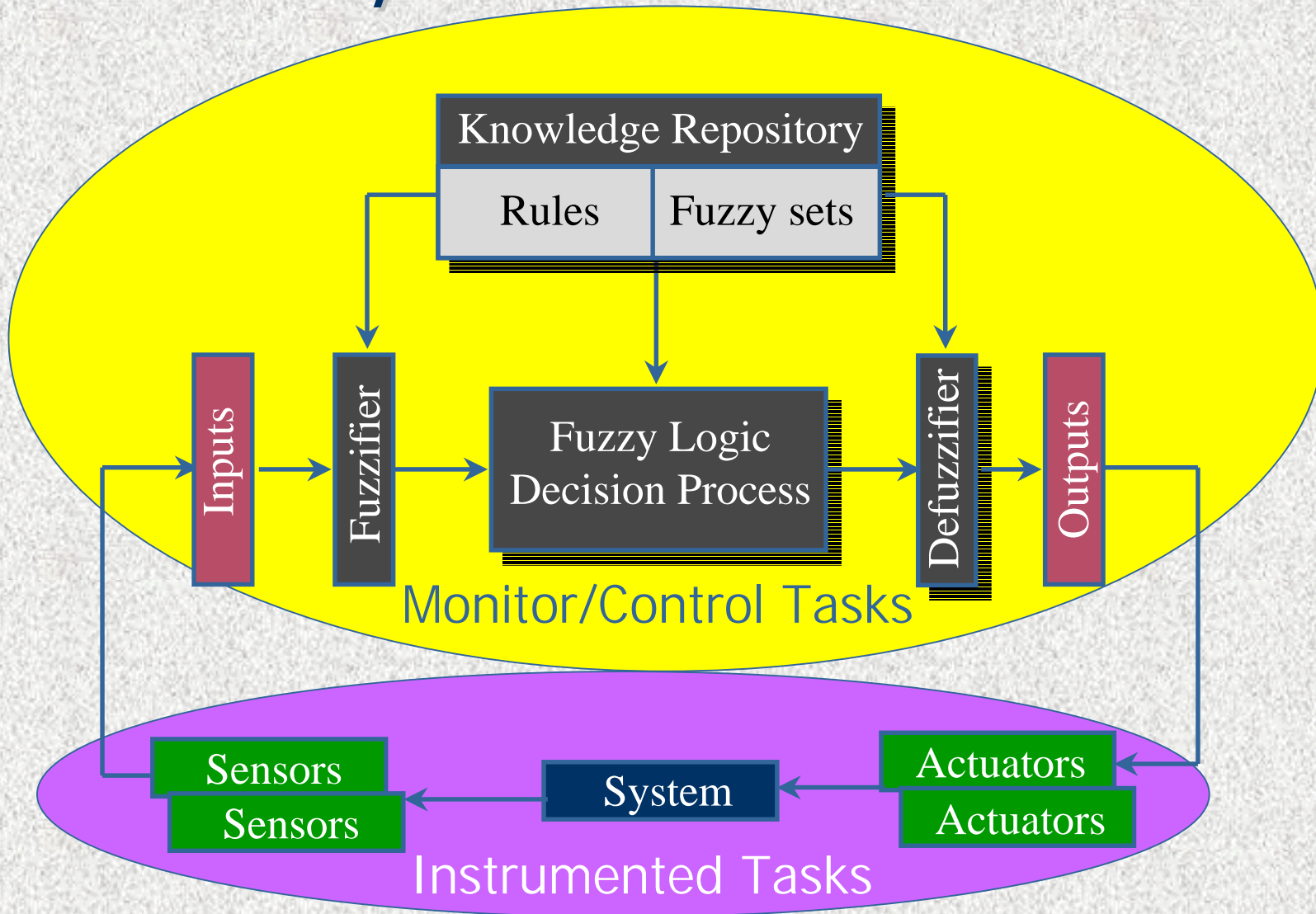


■ Research approach

- monitor resource demands and responses
- select policies based on observed behavior
- implement policy changes locally and globally



Autopilot Decision Process





Autopilot Fuzzy Logic Control

- Humans rely on qualitative rules
 - If it is RAINING, drive SLOWLY
 - If the system is BUSY, backups should be POSTPONED
- Fuzzy logic expresses these rules formally
 - captures human experience
 - supports contradictory statements
 - processes “gray” statements

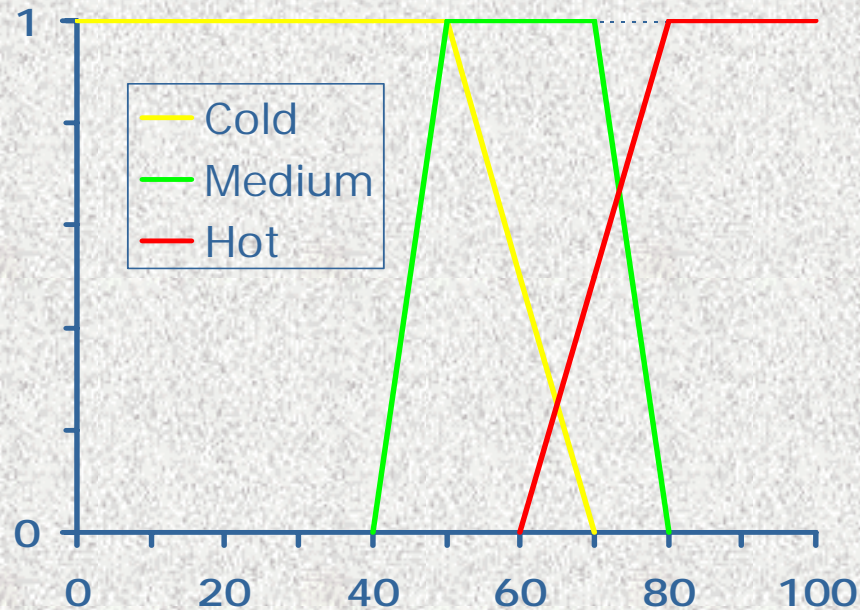


Fuzzy Controller Structure

- Fuzzifier
 - scales and maps input variables to fuzzy sets
- Inference mechanism
 - approximate reasoning block
 - deduces the control action
 - Compositional Rule of Inference (CRI)
- Defuzzifier
 - converts fuzzy output values to control signals
 - several defuzzification methods



Sample Fuzzy Rule Base



```
rulebase FurnaceRules;  
var roomtemp(0,100) {  
  set trapez cold ( 0, 50, 0, 20 );  
  set trapez medium( 50, 70, 10, 10 );  
  set trapez hot ( 80, 100, 20, 0 ); };
```



Sample Fuzzy Rule Base

```
var furnace(0,1) {  
  set triangle off ( 0, 0, 0.1 );  
  set triangle half( 0.5, 0.1, 0.1 );  
  set triangle full( 1, 0.1, 0 ); }  
  
// the rules  
if ( roomtemp == cold ) { furnace = full; }  
if ( roomtemp == medium ) { furnace = half; }  
if ( roomtemp == hot ) { furnace = off; }
```



Sample Fuzzy Logic Control

```
// Fuzzy sensor value
_furnaceRules->roomtemp.value( _sensedTemperature );

// Execute the rule base
_furnaceRules->furnaceRules.base.evaluate_all();

// Defuzzify the outputs
_furnaceRules->furnaceRules.base.defuzzy_all();

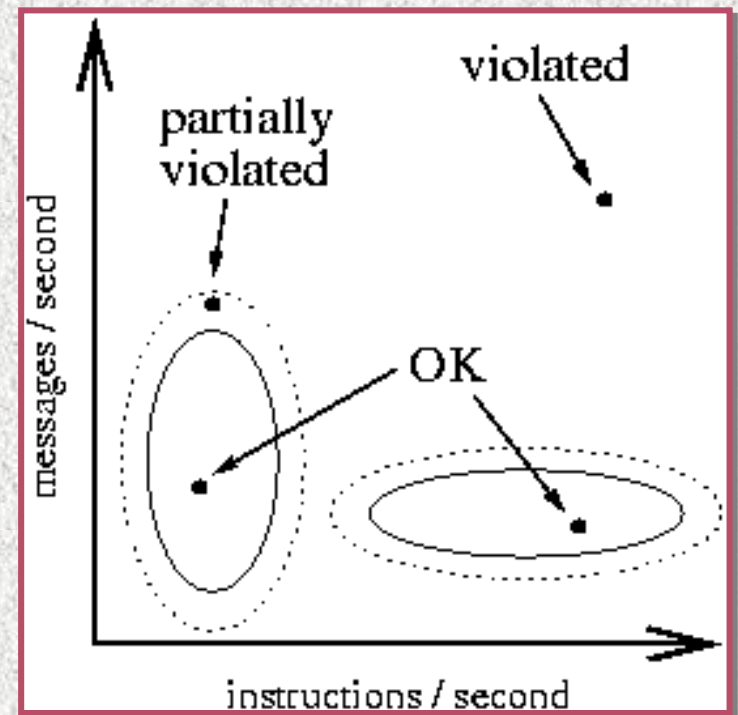
// Update actuator in the instrumented task
_newFurnaceIntensity = _furnaceRules->furnace.value();
_heatActuatorClient->changeValue(
    &_newFurnaceIntensity, 1 );
```



Performance Contracts

- Separation
 - application and system
- Validation
 - runtime measurements
 - testing against projected model
- Violation detection
 - via fuzzy logic rules
 - » nearness to cluster centroids
 - violation is not “black and white”

```
if ( distance == LONG)  
    contract = VIOLATED
```





Application & System Separation

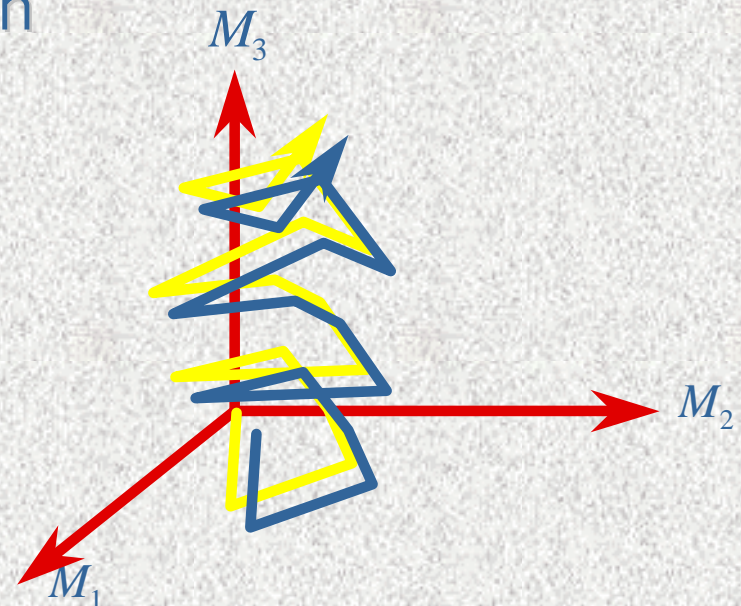
- Application intrinsic behavior
 - independent of execution platform
 - application resource stimuli description

- System behavior
 - description of resource capabilities
 - » MFLOPS, I/O rate, bandwidth, latency, ...
 - anything other than peak is a hybrid
 - » dependent on application stimuli and system capability



Application Signatures

- Application signatures
 - application intrinsic measures
 - » decoupled from resource mapping
 - resource stimuli by application
 - examples
 - » I/O volume/statement
 - » FLOPS/statement
- Temporal evolution
 - metric trajectories in
 - » application space
- *Identification requirements*
 - *relevant tasks and metrics*





Metric Identification

- Challenge
 - local measurement
 - global insight

- Statistical clustering
 - identifies relevant tasks
 - requires periodic reclustering
 - » evolving temporal behavior – consider ScaLAPACK

- Projection pursuit
 - identifies relevant metrics
 - » only a subset are not highly correlated



Clustering and Program Models

- Three basic alternatives
 - functional decomposition
 - single program multiple data (SPMD)
 - data parallel (HPF or HPC++)
- *De facto* equivalence classes
 - functions (different code)
 - SPMD (same code but data dependent)
 - data parallel (same code but data dependent)



Statistical Data Clustering

■ Goals

- identify behavioral equivalence classes
- record data from behavioral representatives
- reduce aggregate data volume
- minimize instrumentation perturbations

■ Approach

- real-time data analysis
- cluster representative logging





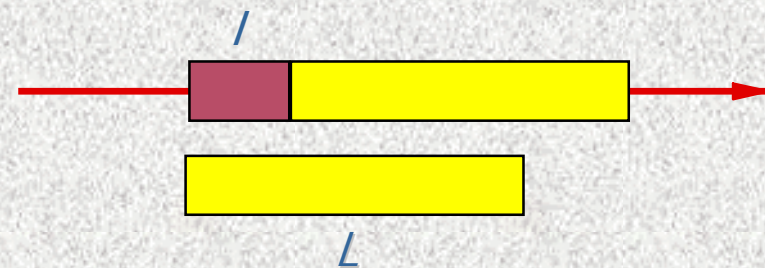
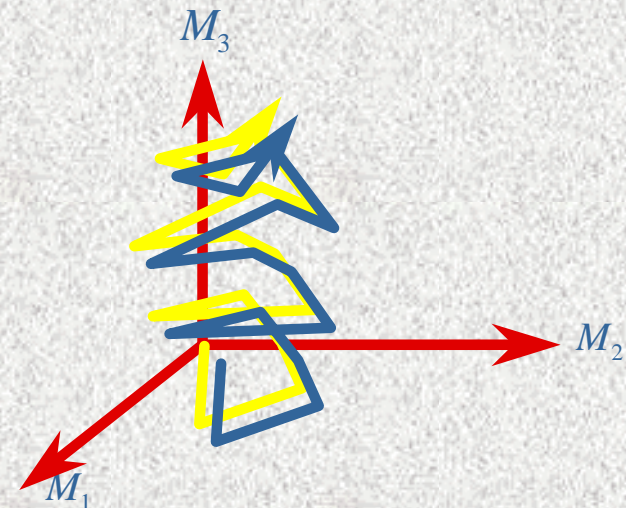
Clustering and Data Smoothing

■ Motivations

- convert events to metrics
- create low-pass filters
- reduce data volume
- mitigate effects of outliers

■ Sliding window averages

- window size L
- window increment $/$
 - » *need not be time*





Temporal Cluster Updates

■ Goals

- cluster periodically
- monitor data dispersion
- recluster when necessary
 - » identify application phase transitions

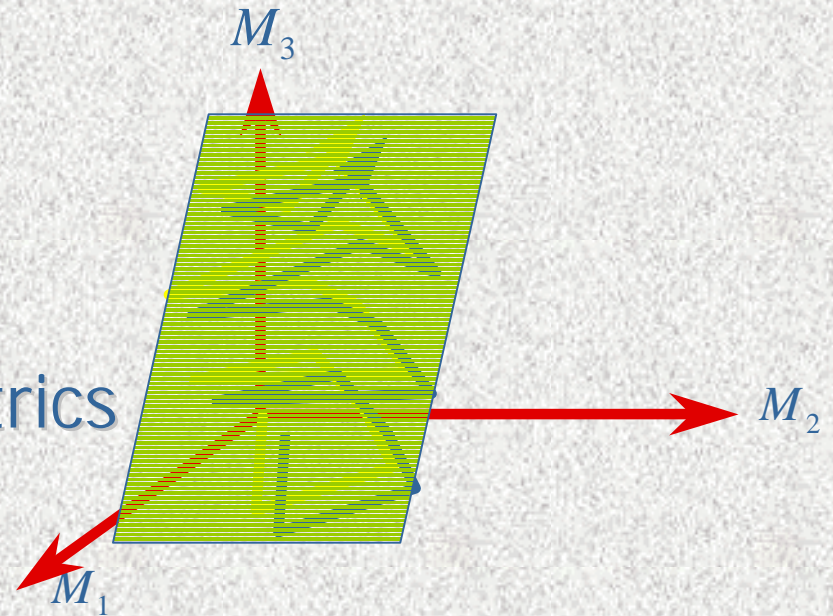
■ Approach

- recompute error at user-specified intervals
- recluster if error exceeds specified threshold



Projection Pursuit

- Statistical clustering
 - reduces the number of data points
 - but *not* their dimensionality (metric count)
- Projection pursuit
 - generalization
 - » principal component analysis
 - identifies “important” metrics





Conceptual Contract Methodology

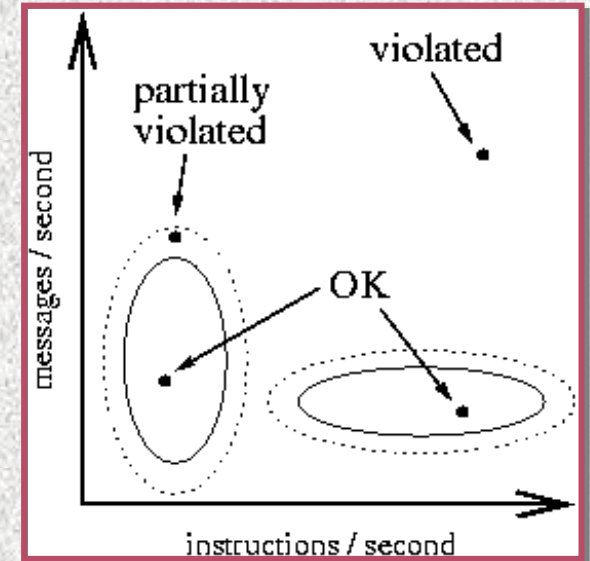
- Combine
 - application characteristics
 - » developer
 - » compile-time analysis
 - » previous application executions
 - resource specifications
 - » Network Weather Service (NWS)
 - » benchmarks, and other data
- Project expected performance
 - using characteristics and specifications (models)
- Measure run-time performance and test contracts
 - using application instrumentation
 - » detect violations
 - » identify causes





Contract Implementation

- Cluster centroids and radii define
 - nominal predicted behavior
 - range of acceptability
- Fuzzy logic rule base specifies
 - standard contracts
 - “confidence” in contract validity
- Execution time measurements provide
 - raw data for contract validation
 - indication of source of violation





Contract Violations

- Multiple possible causes
 - application
 - » different/unexpected inputs
 - » timing dependent behavior
 - system
 - » resource availability and behavior
 - performance models
 - » prediction errors

- Contracts identify possible causes
 - basis for remediation



Trivial Contract Fuzzy Rule Set

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

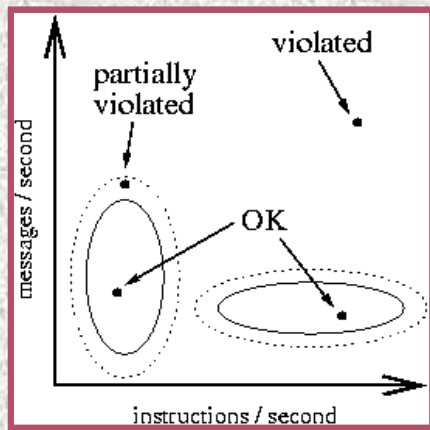
```
if (distanceFromCentroid == LONG)
    {contract = VIOLATED;}
```

- Options for greater sophistication
 - temporal contracts
 - » rule base and/or fuzzy set evolution
 - application/system violation
 - » rules for tracking application metrics
 - » rules for validation resource behavior



Contract Violations

- Rules for tracking application metrics
 - Is observed application signature observed?
 - » Intuitively, is the code in the “application cluster space?”



Intrinsics identify likely application cause

- Rules for validation resource behavior
 - Have the resource rates/capacities been delivered?
 - » Are the single figure of merit values delivered (MFLOPS, BW, ...)?



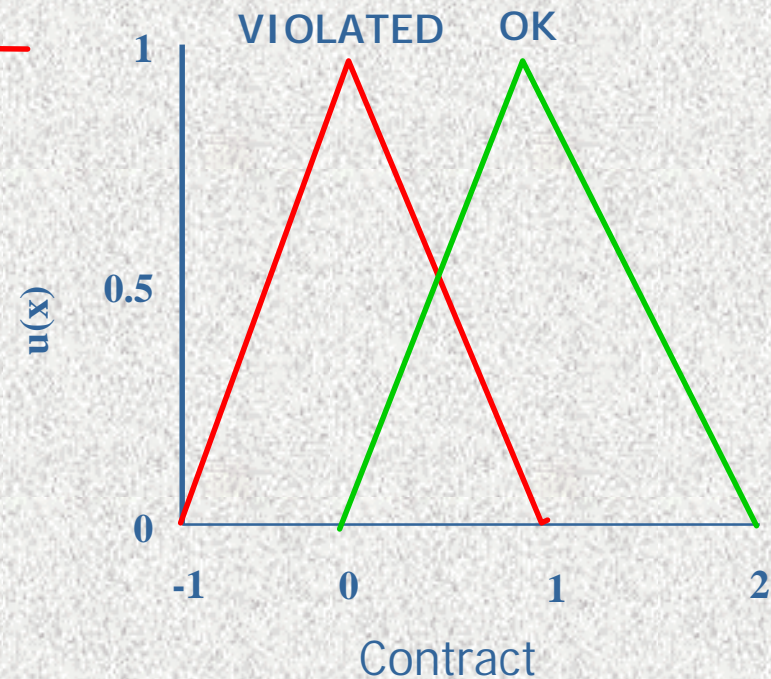
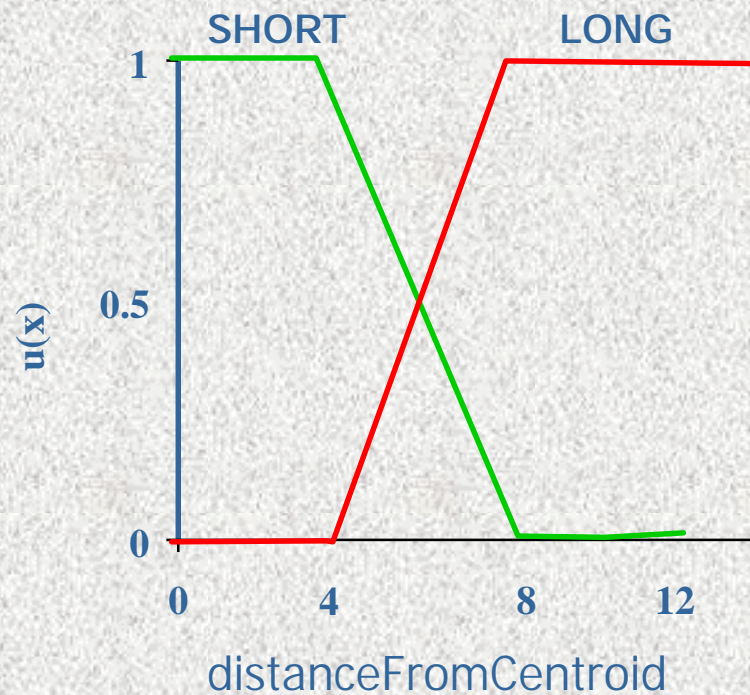
Fuzzy Variable Definitions

```
Rulebase Grads_Contract;
var distanceFromCentroid( 0, 100 )
{
  set trapez  SHORT  ( 0, 4, 1, 4 );
  set trapez  LONG   ( 8, 100, 4, 0 );
}

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



Fuzzy Variables (Membership)





Case Law: A Contract Example

■ Environment

- SC'00 ScaLAPACK demonstration code
- multiple Linux PC clusters
 - » differing network and processor speeds

■ Approach

- capture and cluster application intrinsic metrics
 - » clusters identify behavioral equivalence classes
- project intrinsics onto system metrics
 - » projections yield predicted performance for contracts
- measure performance on various Linux PC clusters
- validate predictions for contract testing



Projection Strategies

- Given
 - application intrinsic behavior
 - resource performance predictions

- Many possible projection strategies
 - parameterized application performance model (best)
 - single figure of merit scaling
 - » peak MFLOPS, bandwidth, and I/O rate
 - » benchmark suite MFLOPS, bandwidth, and I/O rate
 - Linpack, SPEC2000, ...
 - » previous application executions (learning)
 - WAG 😊



ScaLAPACK Test Configuration

- Problem size
 - 2000x2000 matrix
 - panel size (NB) of 64
- Cluster configuration
 - Several different four processor Linux systems
- Data capture
 - loop instrumentation
 - » PAPI hardware counters
 - » Pablo instrumentation via MPI wrappers
 - initial data distribution phase excluded
 - Measurement data from processor 0



Single Figure of Merit Projection

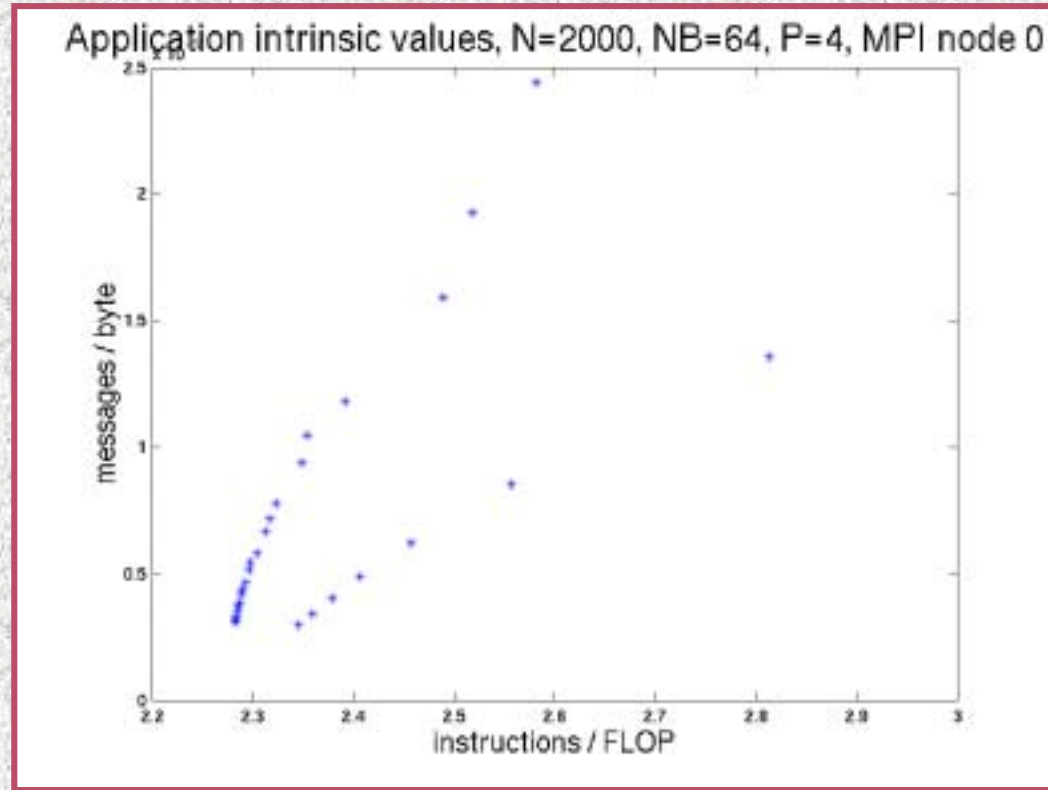
- For each loop iteration
 - measure application intrinsic metrics
 - » instructions/FLOP
 - » messages/byte
 - (N.B. compiler support needed for *statements/FLOP*)

- Obtain projection factors (system characteristics)
 - bytes/second (bandwidth)
 - FLOPS/second (MFLOPS)

- Project onto system specific metrics
 - instructions/second
 - messages/second



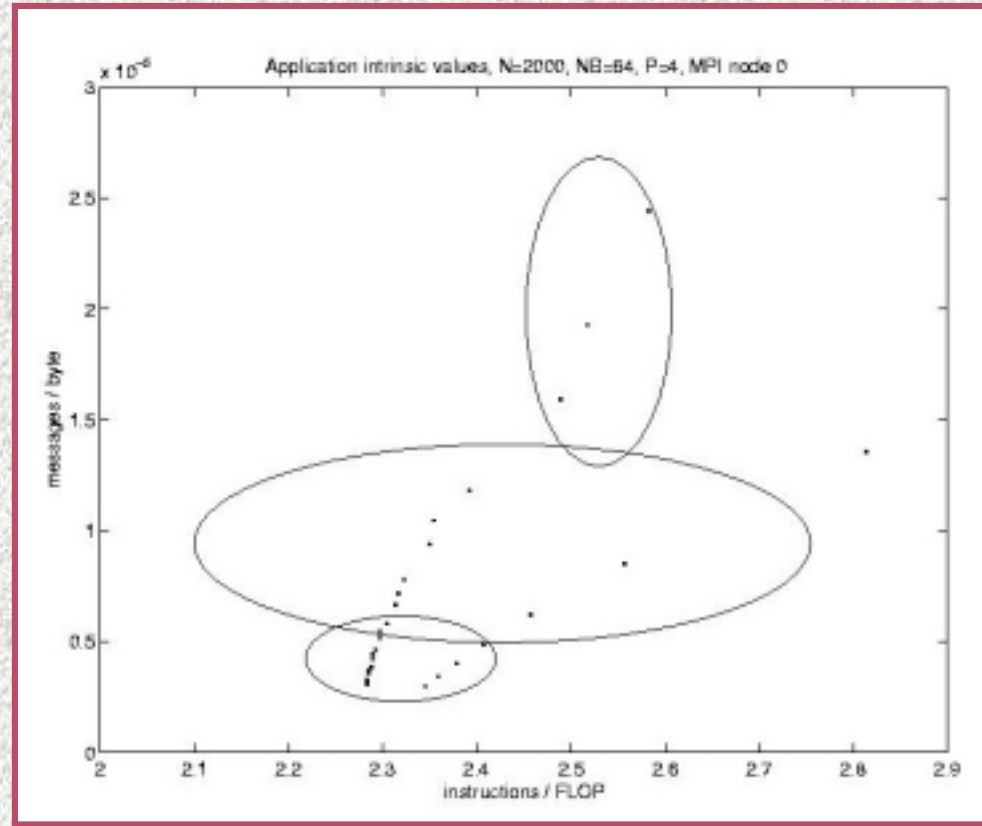
Application Intrinsic Metrics



- System independent
 - *modulo instructions versus statements (need compiler support)*
 - » same binary used for all experiments
 - input to any model (developer, compiler, or regression)



Clustered Intrinsic Metrics



- No smoothing applied – simple test case
– metric trajectories lost



Three Cluster Environments

- Major cluster
 - 266 MHz Pentium II
 - 100Mb Ethernet
- Opus cluster
 - 450 MHz Pentium II
 - Myrinet
- Mixed cluster
 - amajor, bmajor, torc7.utk, torc8.utk
 - 2 x 266 MHz Pentium II
 - 2 x 550 MHz Pentium III
 - Internet



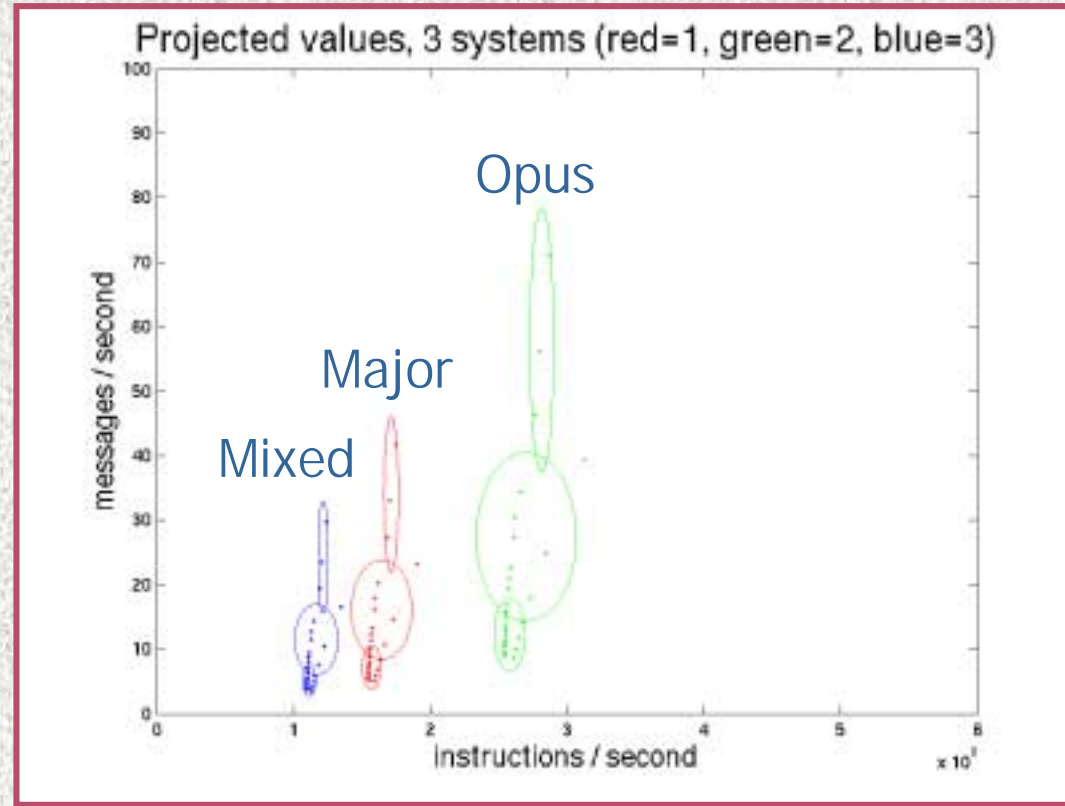


Single Figure of Merit Projection

- Simple test case
 - measure actual application performance (ScaLAPACK)
 - project using measured application scale factors
 - » Mb/second and MF/second
- Major cluster
 - 13.7 Mb/s and 67.8 MF/s
- Opus cluster
 - 23.3 Mb/s and 111.3 MF/s
- Mixed cluster
 - 9.76 Mb/s and 48.1 MF/s



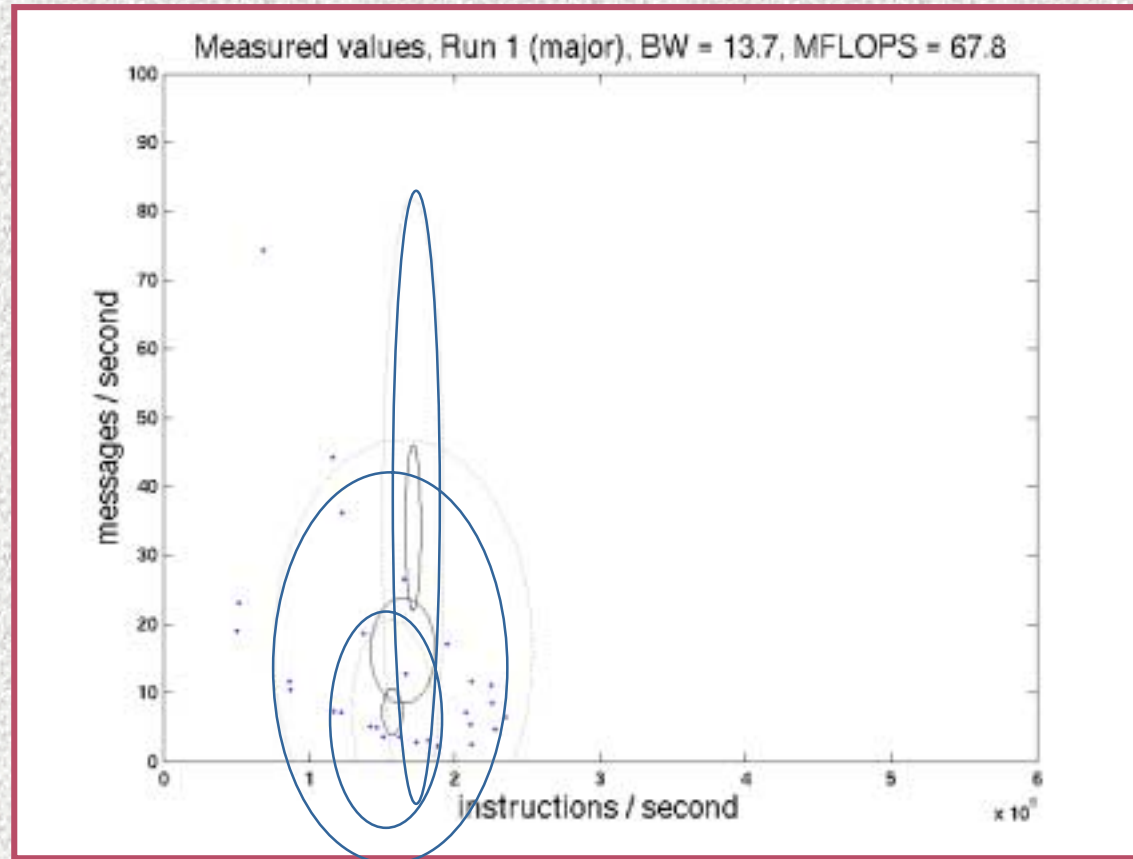
Projected Behavior



- Expected behavior
 - *if single figure of merit were accurate*
 - *actual behavior is different*



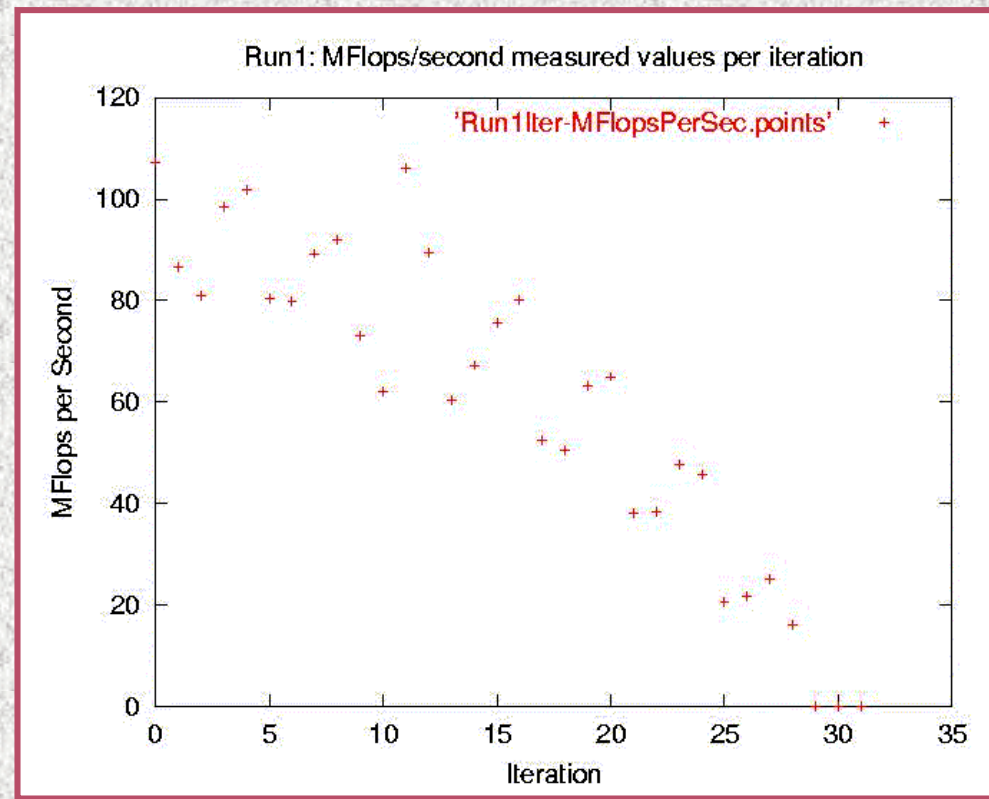
Experimental Error



- Scaled centroids
 - margin of error for robust contracts



Error Sources

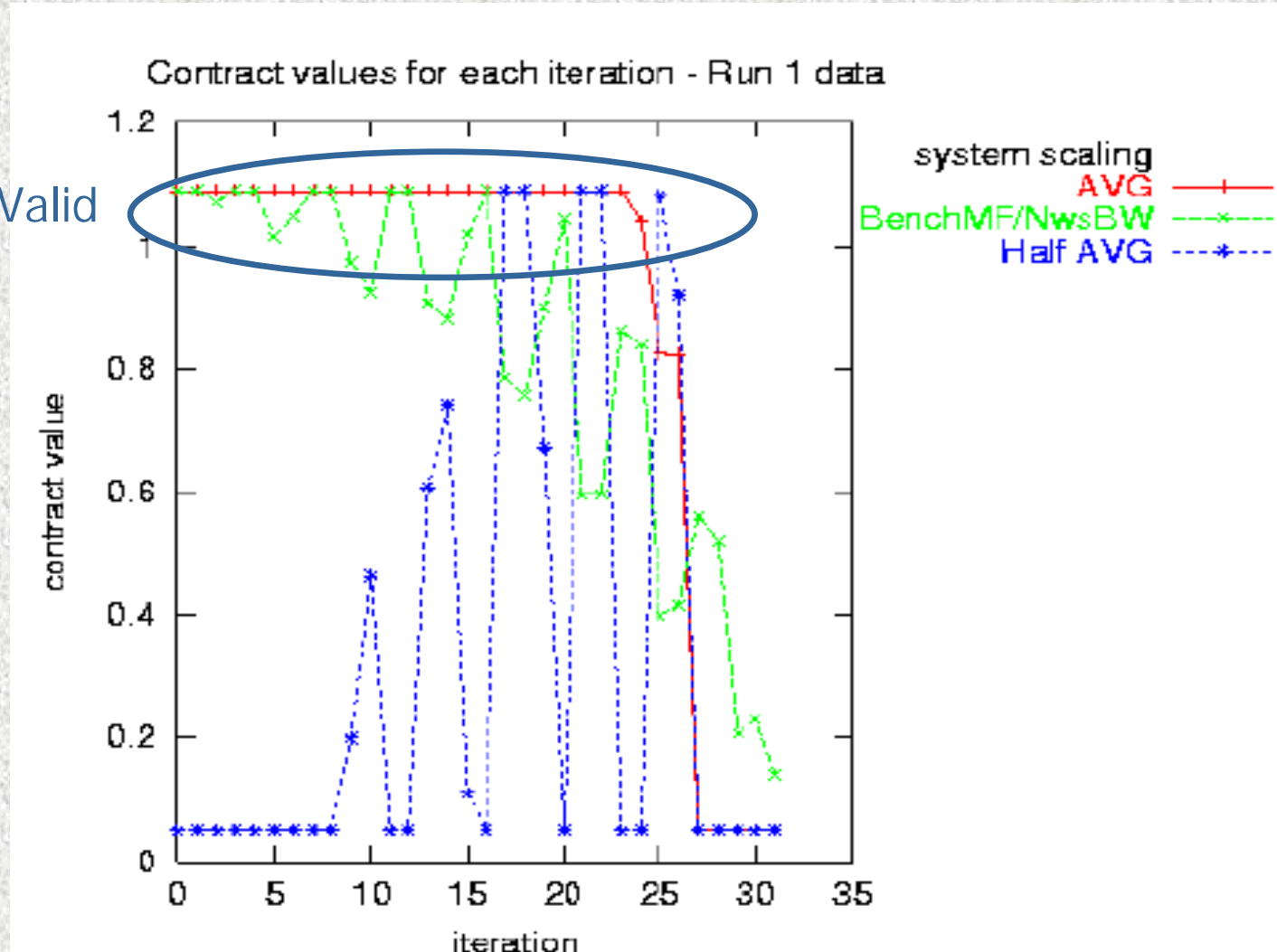


- Errors due to temporal evolution
- Solution
 - cluster and project periodically – identifying “phases”



Contract Values

Contract Valid





Configurable Object Program

- Representation of the application
 - dynamic reconfiguration and optimization
 - » for distributed targets
 - includes
 - » program intermediate code
 - » annotations from the compiler (reconfiguration strategy)
 - » historical information (run profile to now)

- Reconfiguration strategies
 - aggregation of data regions (submeshes)
 - aggregation of tasks
 - definition/redefinition of parameters
 - » used for algorithm selection



More Powerful Application Models

- Many possibilities
 - Artificial Neural Nets (ANNs)
 - Hidden Markov Models (HMMs)
 - time series models (e.g., ARIMA)
 - compile-time symbolic models

- Rationale
 - application behavioral classification
 - » temporal and state transition



Classification Examples

■ ANNs

- I/O and communication classification
 - » qualitative description of behavior
 - » E.g., strided, large/small

■ HMMs

- complex, non-qualitative classifications
 - » probabilistic predictions of behavior
- E.g., irregular spatial I/O patterns

■ Time series

- Inter-operation intervals
 - » Need not be time, could be statements