PICS - a Performance-analysis-based Introspective Control System to Steer Parallel Applications

Yanhua Sun, Jonathan Lifflander, Laxmikant V. Kalé

Parallel Programming Laboratory
University of Illinois at Urbana-Champaign

sun51@illinois.edu

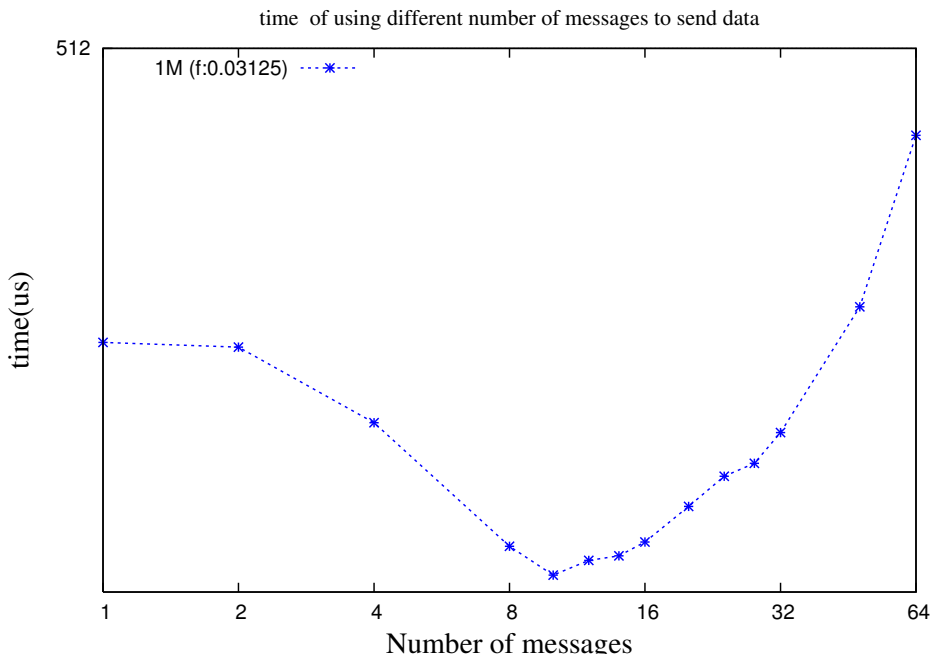
June 10, 2014

Motivation

1. Modern parallel computer systems are becoming extremely complex due to network topologies, hierarchical storage systems, heterogeneous processing units, etc.
2. Obtaining the best performance is challenging.
3. Moreover, multiple configurations for the same application.

Motivation

1. Modern parallel computer systems are becoming extremely complex due to network topologies, hierarchical storage systems, heterogeneous processing units, etc.
2. Obtaining the best performance is challenging.
3. Moreover, multiple configurations for the same application.



Introspection and Adaptivity

General Observation

Configurations of tunable parameters in the runtime system and applications significantly affect the performance.

Top Ten Exascale Research Challenges in DOE Report

"Introspection and automatic adaptation is listed as significant research topic to achieve the performance goal on exascale computers."

Introspection and Adaptivity

General Observation

Configurations of tunable parameters in the runtime system and applications significantly affect the performance.

Top Ten Exascale Research Challenges in DOE Report

"Introspection and automatic adaptation is listed as significant research topic to achieve the performance goal on exascale computers."

Statement

This work addresses the problem of how to improve both parallel programming productivity and performance by letting applications/runtime expose tunable parameters and letting the control system figure out the optimal configurations of these parameters.

Related work

- Autotuning frameworks : generate multiple implementations (FFTW)
- Autopilot[Ribler et al.(1998)]: fuzzy logic rules, grid applications, resource managements
- MATE [Morajko 2006] : fully automatic tuning, performance model
- Active Harmony[Chung and Hollingsworth(2006)] : heuristic algorithms
- SEEC: A General and Extensible Framework for Self-Aware Computing[Henry Homann (2010,2011,2013)]

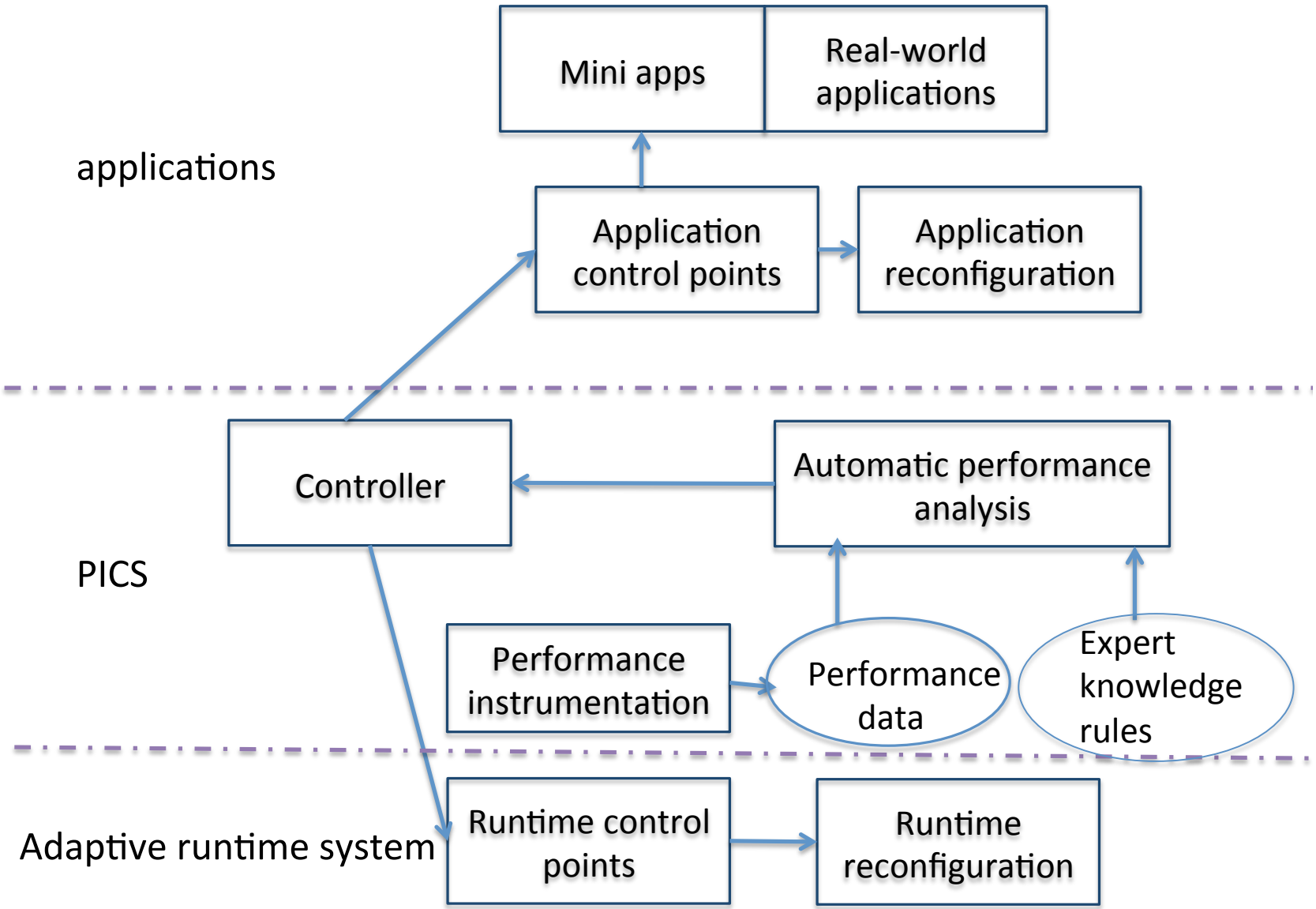
Our Approach

- HPC applications on large scale
- Not rely on performance models
- Richer set of tunable parameters due to the powerful intelligent runtime system
- Not only application configurations are tunned, but also the runtime system itself
- Automatic performance analysis accelerates steering

Outline

- Overview of PICS framework
- Control points in the runtime system and applications
- Automatic performance analysis to accelerate steering
- APIs implemented in Charm++
- Results of benchmarks and applications

Overview of PICS framework



Control Points

Control points

Control points are tunable parameters for application and runtime to interact with control system. First proposed in Dooley's research.

1. Name, Values : default, min, max
2. Movement unit: $+1, \times 2$
3. Effects, directions
   - Degree of parallelism
   - Grainsize
   - Priority
   - Memory usage
   - GPU load
   - Message size
   - Number of messages
   - other effects

Application and Runtime Control Points

Application

1. Application specific control points provided by users
2. Applications should be able to reconfigure to use new values

Runtime

1. Traditionally, configurations for the runtime system do not change
2. Configurations for the runtime system itself should be tunable

1. Registered by runtime itself
2. Requires no change from applications
3. Affect all applications

Observe Program Behaviors

- Record all events
  - Events : begin idle, end idle
  - Functions: name, begin execution, end execution
  - Communication : message creation, size, source/destination
  - Hardware counters
- Module link, no source code modification
- Performance summary data

Automatically Analyze the Performance

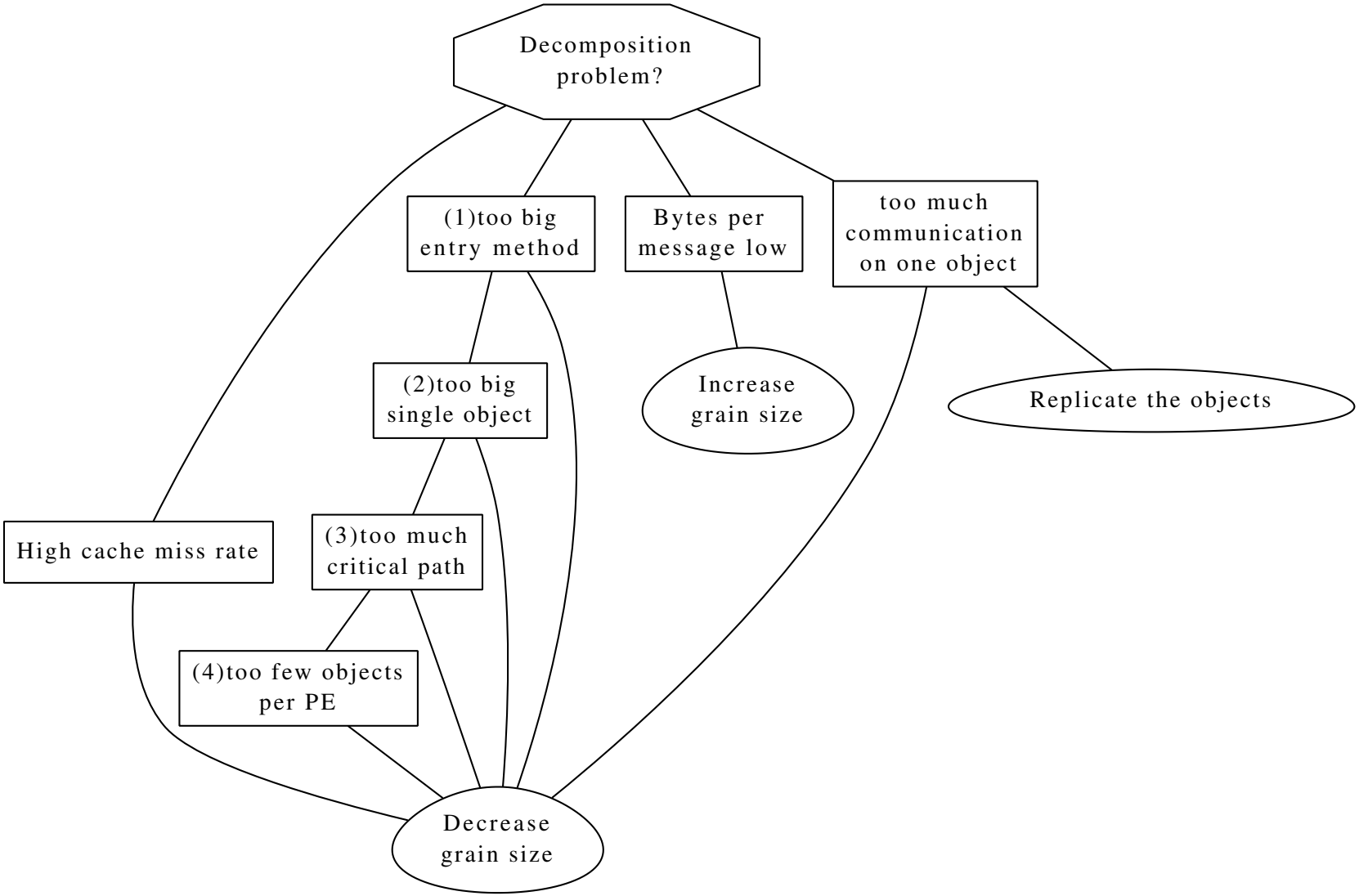
Many control points are registered. How to reduce the search space?

Automatically Analyze the Performance

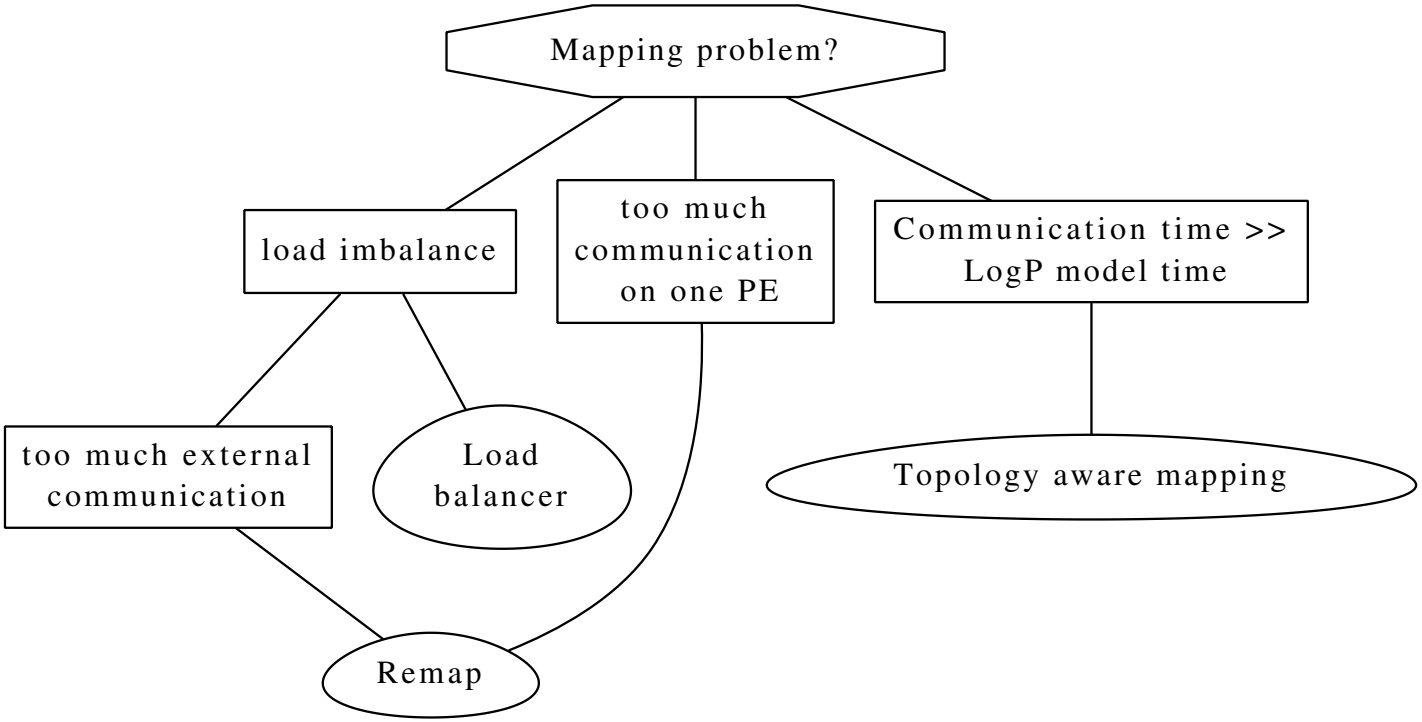
Many control points are registered. How to reduce the search space?
Performance Analysis - Identify Program Problems

- Decomposition
- Mapping
- Scheduling

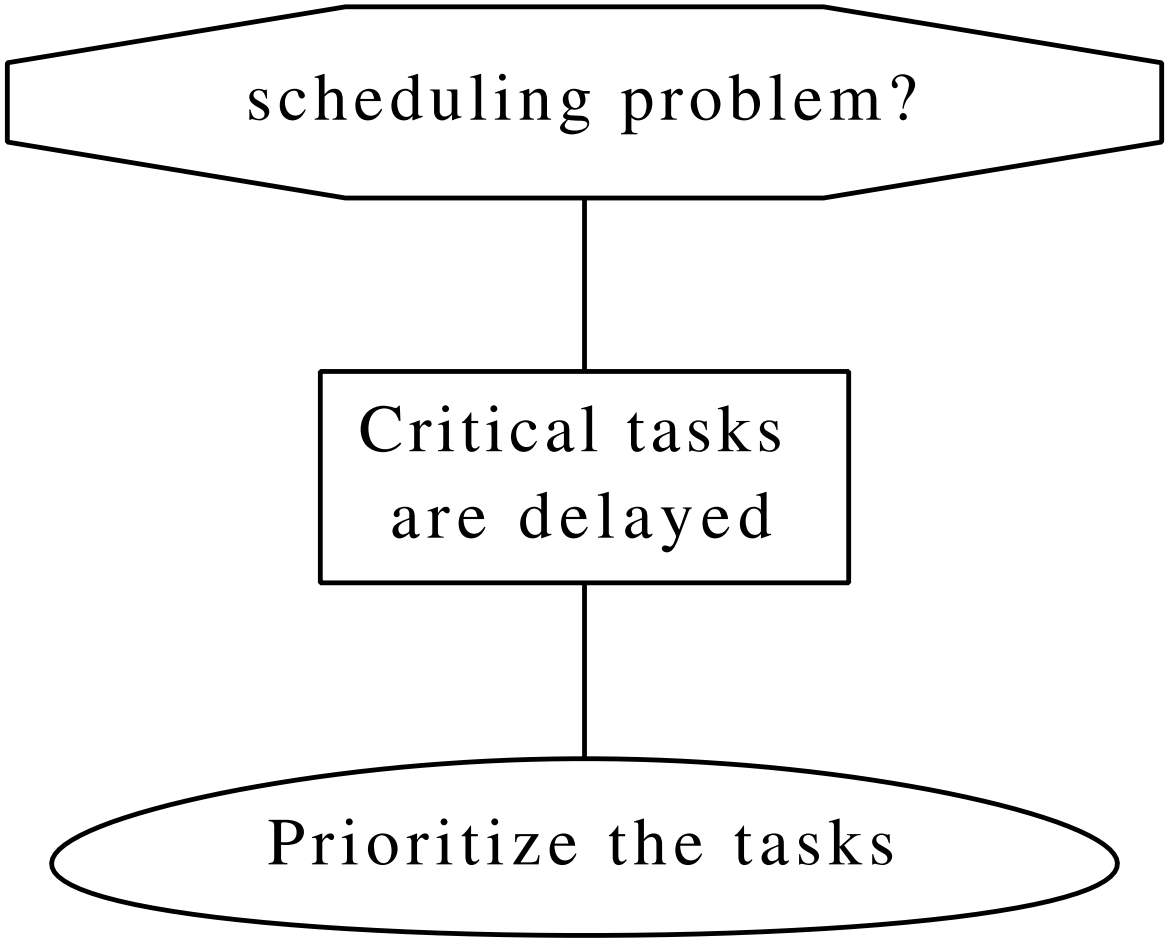
Decomposition Characteristics



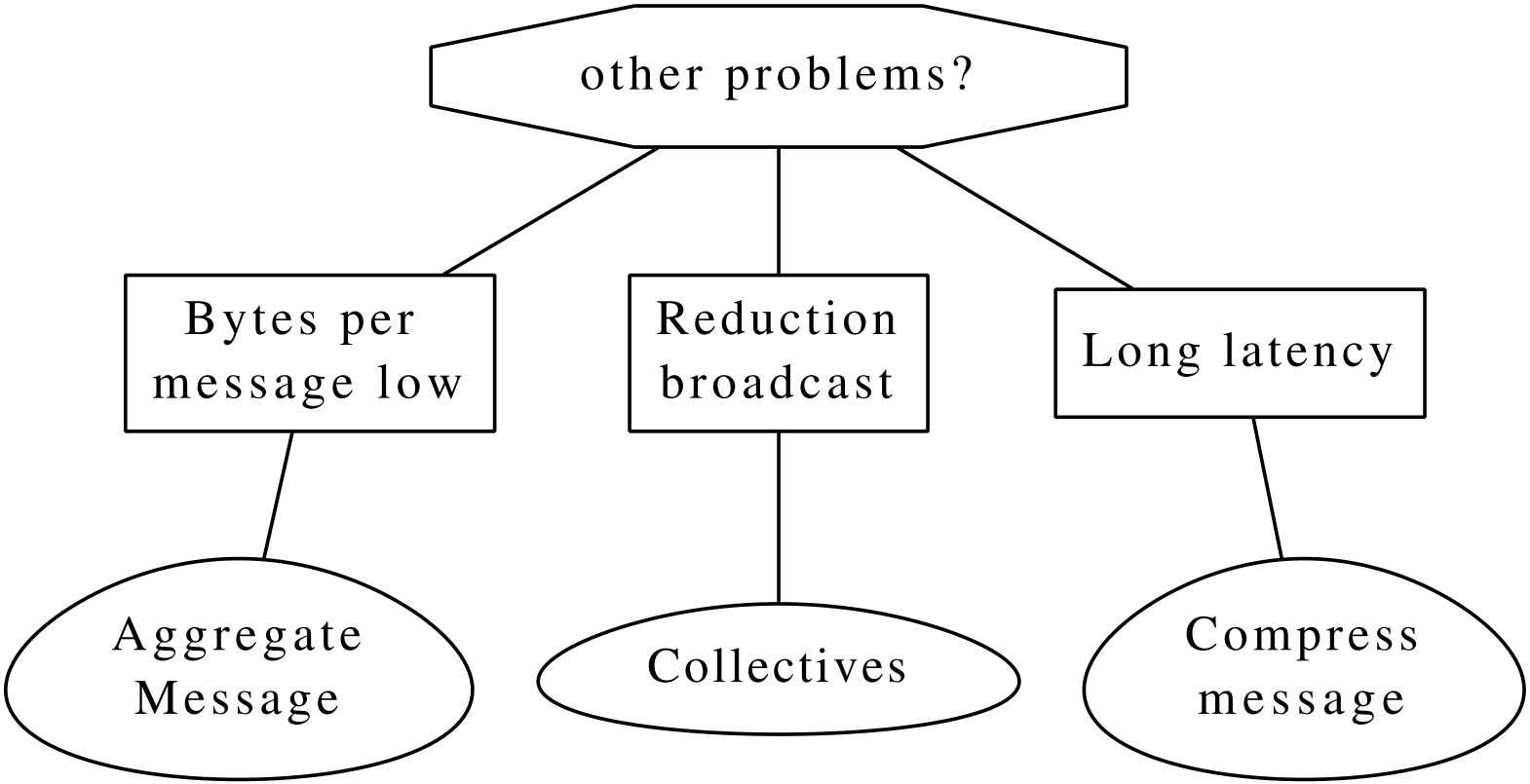
Mapping Characteristics



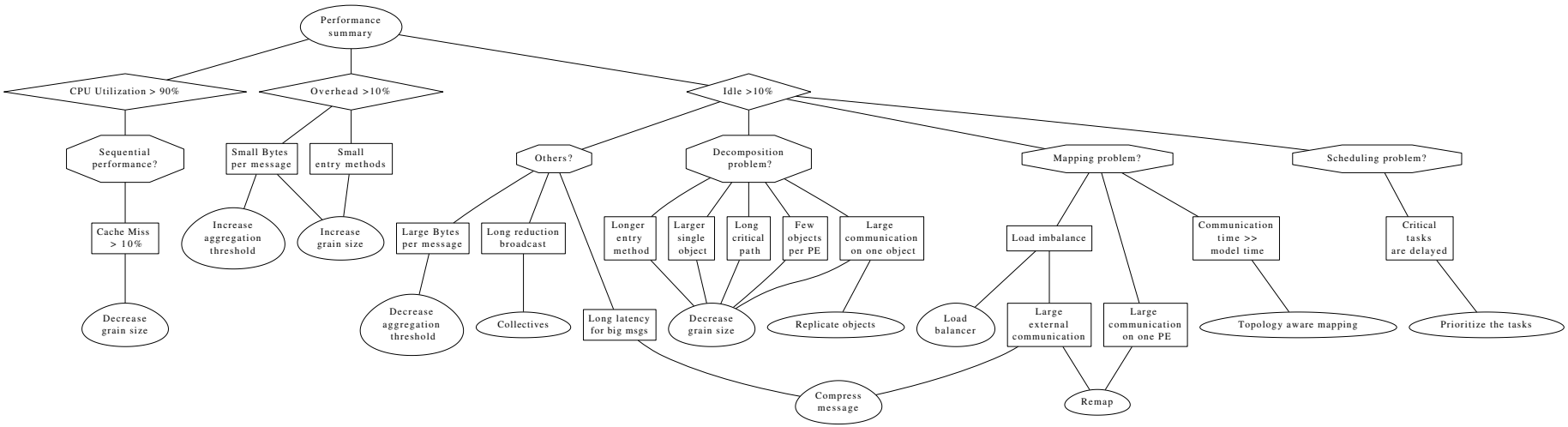
Scheduling Characteristics



Other Characteristics



Correlate Performance with Control Points



- One box can have multiple children
- One egg can have multiple parents

Correlate Performance with Control Points

Traverse the tree using the performance summary results

- performance results $\Rightarrow$ solutions
- solution $\Rightarrow$ effect of control points
- What control points to tune, in which direction!
- How much?
  - grainsize : $\frac{MaxObjLoad}{AvgLoad}$
- Feed results into the control points database

Control System APIs

Implemented in Charm++, over-decomposition, asynchronous, message-driven model. (http://charm.cs.uiuc.edu/)

```
typedef struct ControlPoint_t
{
    char     name[30];
    enum     TP_DATATYPE datatype;
    double   defaultValue;
    double   currentValue;
    double   minValue;
    double   maxValue;
    double   bestValue;
    double   moveUnit;
    int      moveOP;
    int      effect;
    int      effectDirection;
    int      strategy;
    int      entryEP;
    int      objectID;
}ControlPoint;
```

APIs for applications

```
void registerControlPoint(ControlPoint *tp);

void startStep();
void endStep();

double getTunedParameter(const char *name, bool *valid);
```

Experimental Results of Benchmarks and Applications

1. Control points
2. Performance problems
3. Bluegene/Q machine, Cray XE6 machine

Tuning Message Pipeline

- Control point: number of pipeline messages

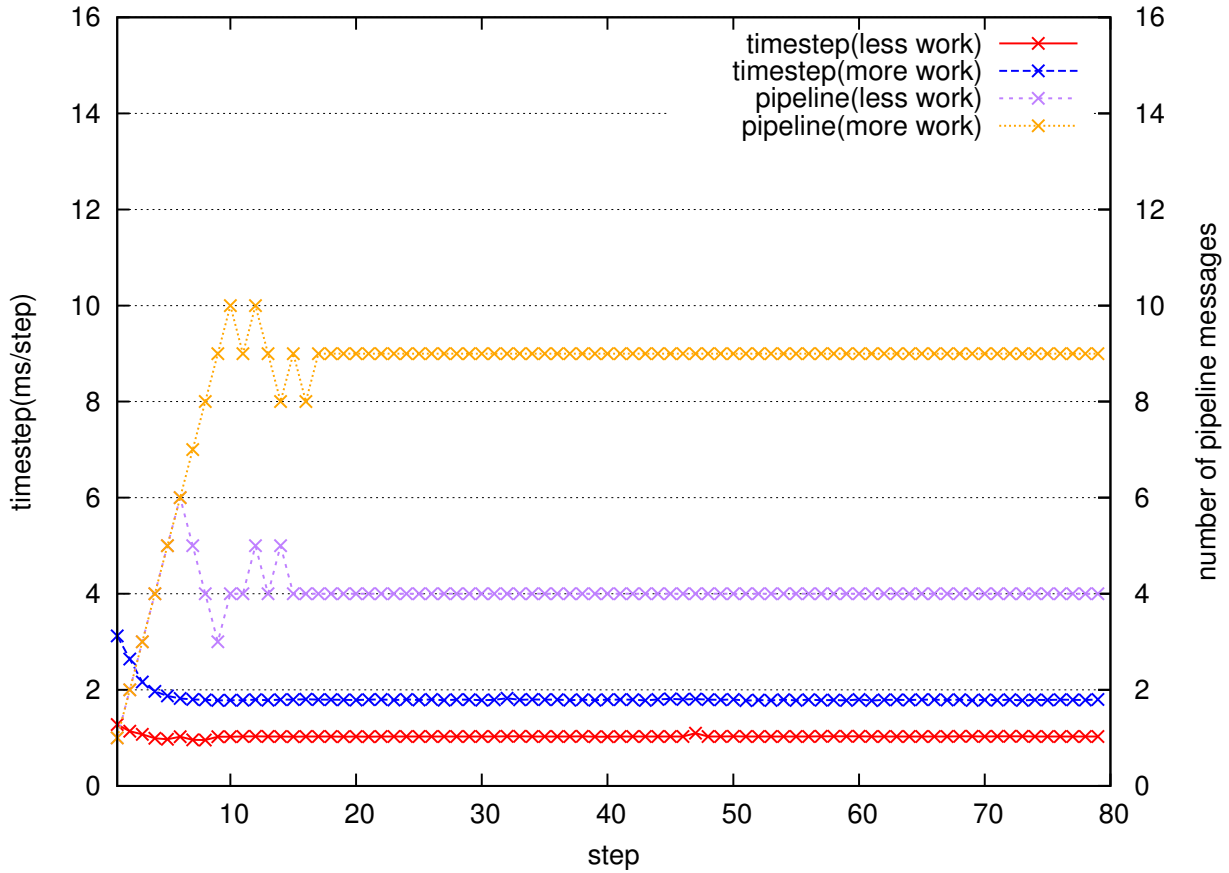


Figure: Tuning the number of pipeline messages

Communication Bottleneck in ChaNGa

- Control points: number of mirrors

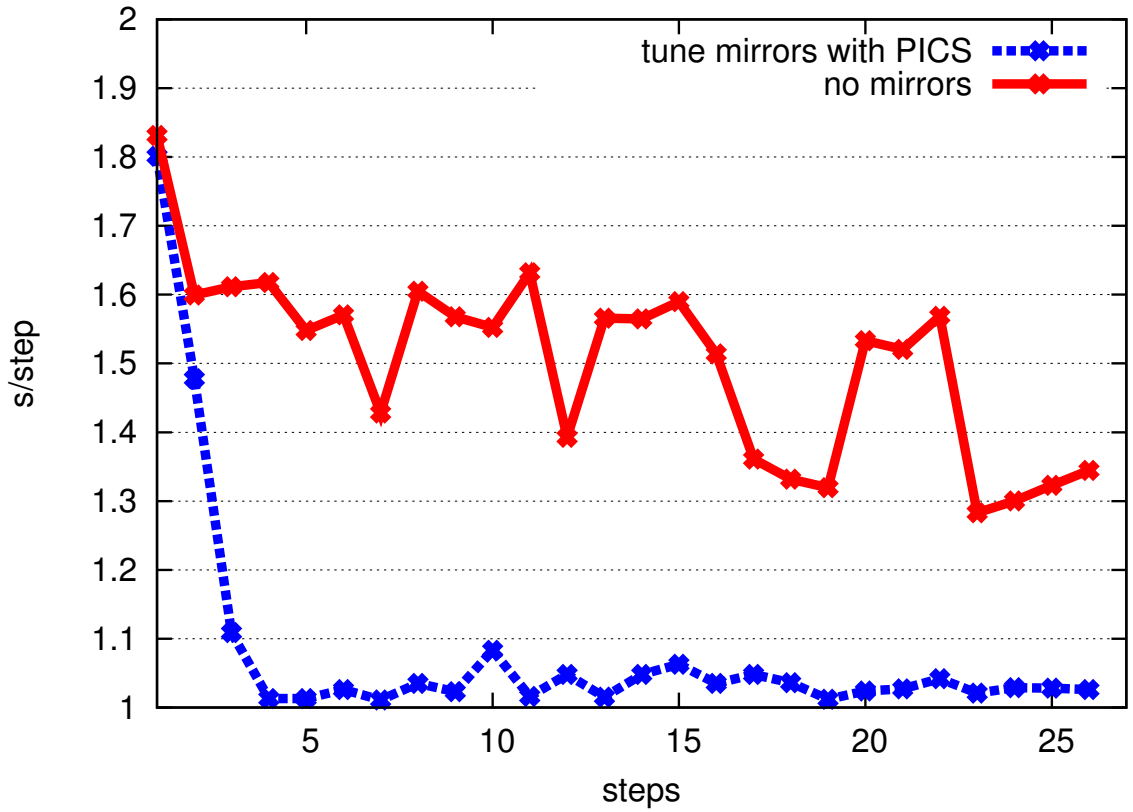


Figure: Time cost of calculating gravity for various mirrors and no mirror on 16k cores on Blue Gene/Q

Message Compression

- Control points: compression algorithm for each type message
- Runtime control points

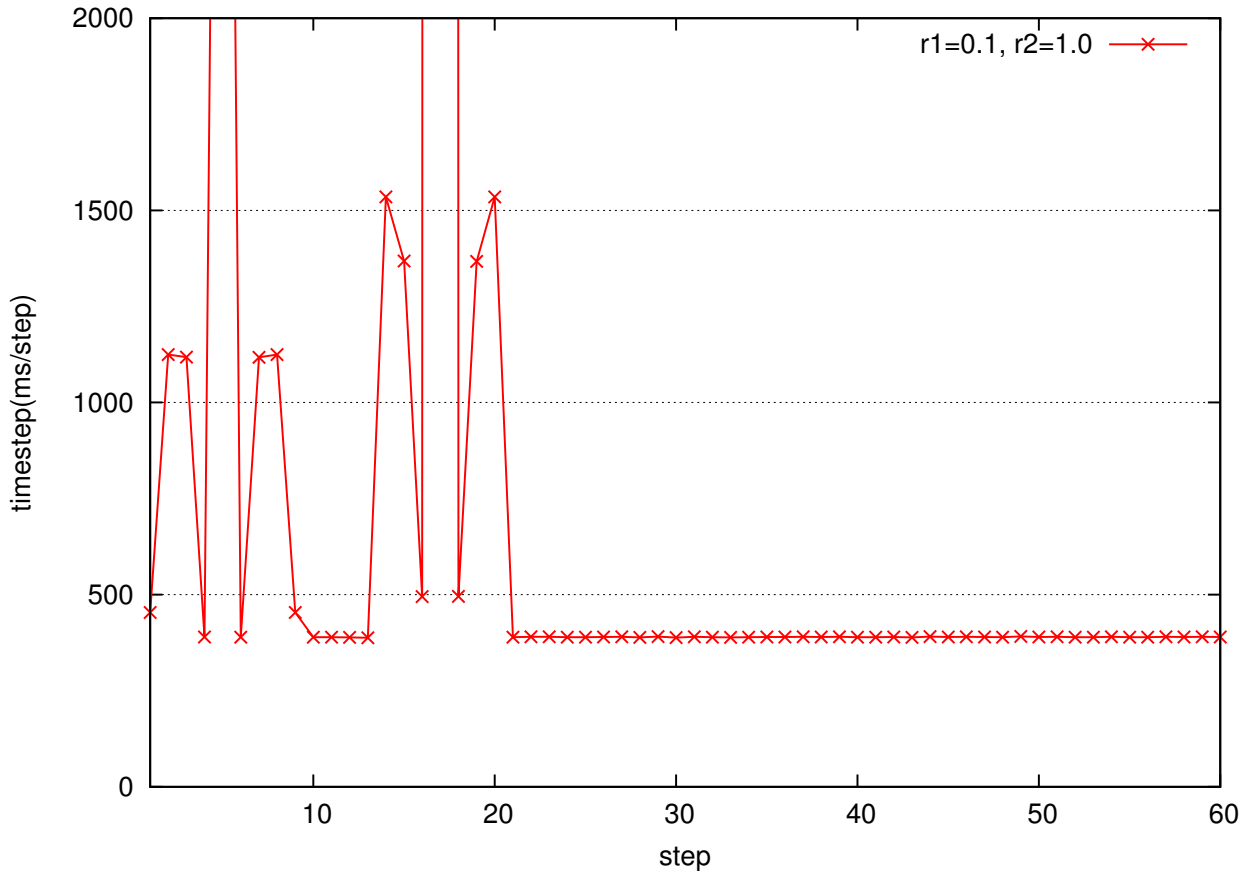


Figure: Steering the compression algorithm for all-to-all benchmark

Jacobi3d Performance Steering

- Control Points: sub-block size in each dimension
- Three control points
- Cache miss rate, high idle suggest decreasing sub-block size
- Overhead

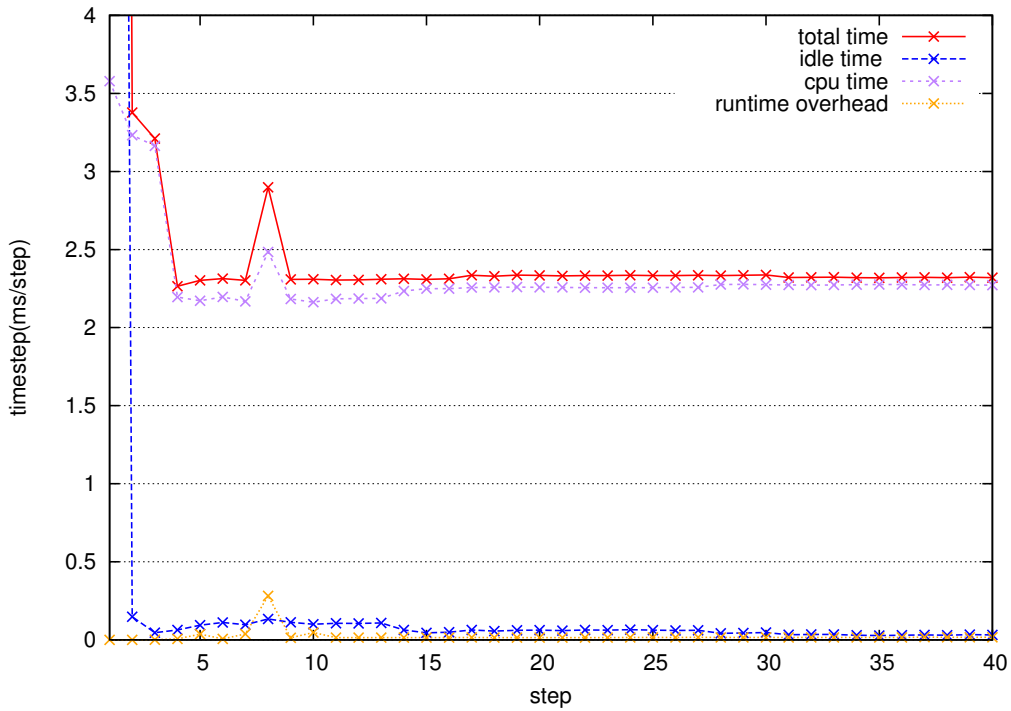


Figure: Jacobi3d performance steering on 64 cores for problem of 1024*1024*1024

Conclusion

- Introspective control system is required to improve productivity and performance
- Automatic performance analysis helps guide performance steering
- Steering both runtime system and applications are important
- Implemented the system based on Charm++ programming model

Acknowledgment

This work was supported in part by NIH Grant 9P41GM104601, Center for Macromolecular Modeling and Bioinformatics. It was also supported in part by DOE DE-AC02-06CH11357 Argo Project. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory.