

---

# Toward Software Systems for Constructing Adaptive Grid Programs

**John Mellor-Crummey**  
Department of Computer Science  
Rice University

[http://hipersoft.rice.edu/stc\\_site\\_visit/talks/adaptive.pdf](http://hipersoft.rice.edu/stc_site_visit/talks/adaptive.pdf)

# A Vision for the Future ...

---

## Grid applications

- Help select appropriate resources
- Continuously monitor their own performance
- Adapt to changing resource availability
- Adapt to changing network conditions

# Work in Progress ...

---

## A framework for adaptive grid computing

- **Application Manager**
  - authority for application specific knowledge and control
- **Resource Negotiator**
  - broker that identifies promising resource sets for application
- **Contract Monitor**
  - agent that checks progress of a running application

# A Missing Link

---

## Support for adaptive applications

- Performance modeling
- Support for reconfiguration
- Latency tolerance
- Fault tolerance
  - primarily checkpoint/restart

# Key Performance Modeling Challenges

---

- Creating accurate parameterized models for resource selection
  - select an appropriately sized collection of suitable resources
  - function of architectural characteristics and problem size
- Creating models for discovering performance problems
- Creating models that guide response to performance problems
  - is one or more resources not meeting application needs?
  - can the problem can be fixed by adjusting the execution?
    - adjust for greater latency tolerance, compress communication
  - are resources delivering promised performance?
  - is there one or more resources that need to be changed?
  - does the application need to contract or migrate?

# Basis for Performance Models

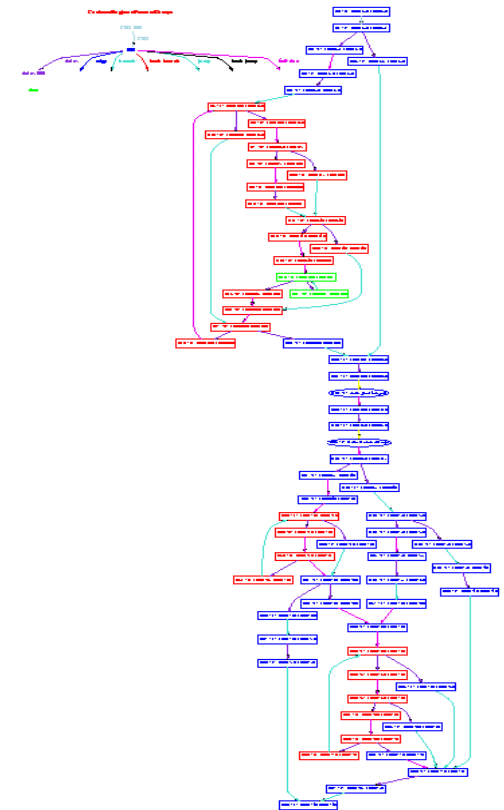
---

- **Static analysis**
  - is the work deterministic or data dependent
  - what is the instruction mix in each basic block
- **Results from sample executions**
  - what are the characteristics of the computation
    - which are the key computational kernels
    - working set size, cache utilization
  - what are the patterns and characteristics of the communication
    - msg size, frequency
  - are the characteristics stable throughout execution
  - what is the critical path
- **On-the fly monitoring**
  - construct a new model based on observation of part of an execution

# Modeling Node Performance

## Understanding architectural impact on performance

- Construct a parameterized model for each scope
  - recover loop nesting structure from an application binary
    - construct control flow graph from binary
    - use interval analysis to recover loop nesting
  - represent performance as a function of key characteristics
    - architectural parameters
    - code characteristics and behavior
      - mix and count of instructions
      - cache utilization
- Compose an aggregate model from loop-level models



# Modeling Parallel Performance

---

## Detailed performance models for black box programs

- The model: partially ordered execution graph (DAG)
  - nodes represent synchronization points
  - 2 flavors of edges
    - processor computation edges
    - synchronization edges
- Construction: binary rewriting to record edge costs
  - identify and instrument communication
    - synchronization edges: communication partners, msg volume
  - rewrite binary to collect basic block histograms between synchronization points
    - processor edges: amount and characteristics of computation

# Generalizing Measured Performance

---

## From sample executions to parameterizable models

- Record communication and computation in one or more executions
  - also measure characteristics with hardware performance counters
- Fit quantitative data using truncated power splines
  - piecewise polynomials joined at “knots”
- Compose performance models from components
- Important characteristics
  - quantitative
    - how does work depend on problem size and architectural parameters?
    - how is work between different synchronization points related?
    - how does the frequency of synchronization evolve?
  - qualitative
    - how do executions with different problem sizes compare?
    - is synchronization local or global?

# Support for Reconfiguration

---

- **Problems**
  - changing computation needs
  - changing resources
- **Goals**
  - dynamically balance load
  - expand, contract or migrate as appropriate
- **Preliminary approach**
  - minor adaptation
    - organize computation into minimum granularity “blocks”
      - migrate blocks to balance load
    - spawn new subtasks on new nodes
  - migration
    - initially through checkpoint/restart

# Latency and Asynchrony Tolerance

---

**Overlap communication with computation to  
minimize serialization**

- **Strategies**
  - eager send
  - overpartitioning for communication pipelining
  - reducing communication frequency
    - partially replicating computation to reduce communication frequency
    - time skewing for iterative computations
      - can reduce communication frequency without replicating computation by tiling “space-time”

# Need for Collaboration

---

- Motivation for component technologies has come from collaborations with application groups
  - impediments to grid computing become opportunities for automation
- New ideas will grow out of continued collaborations
- Exploring and evaluating technologies for program preparation requires a sophisticated software infrastructure
  - team with large spectrum of talents
  - long-term focus