# Adaptive Applications

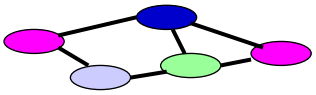
- Last Time
  - » Discussion and Project Plans
  - » Before that: Dynamic Information and its Use
- Today
  - » Adaptation by Applications
  - » What do you need to know?
  - » How to do it well
- Reminders/Announcements
  - » Revised Project Proposals Due Monday, April 25

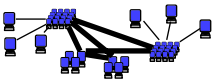


# Today's Readings

- New Grid Scheduling and Rescheduling Methods in the GrADS Project, NSF Next Generation Software Workshop, International Parallel and Distributed Processing Symposium, Santa Fe, April 2004.
  - » O. Sievert, H. Casanova, Policies for Swapping MPI Processes, Proceedings of HPDC-12, the Symposium on High Performance and Distributed Computing, Seattle, June 2003.
  - » Sathish S. Vadhiyar and Jack J. Dongarra, A Performance Oriented Migration Framework for the Grid, CCGrid, IEEE Computing Clusters and the Grid, CCGrid 2003, Tokyo, Japan, May 12-15, 2003.
- The GrADS Project: Software Support for High-Level Grid Application Development, International Journal of High Performance Computing Applications, Winter 2001 (Volume 15, Number 4), pp. 327-344. (General Approach)

# Motivation





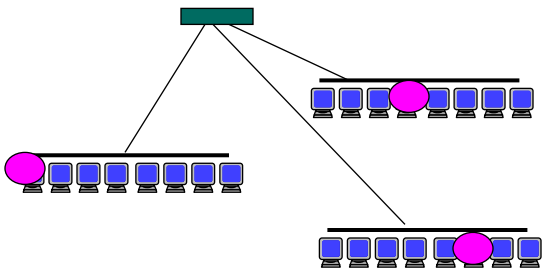
**Resource-Aware**
Describe Resources
Options, Select Resources
Run

**Dynamic Adaptive**
Collect Runtime Info
Consider/Decide
to Change Resources
Select Resources
Run

- Resource-aware Applications? Dynamic Adaptive Applications?
- What information can be obtained to make intelligent decisions?
  - » Historical state? Current state? Future State?
- How does an application decide when to adapt?

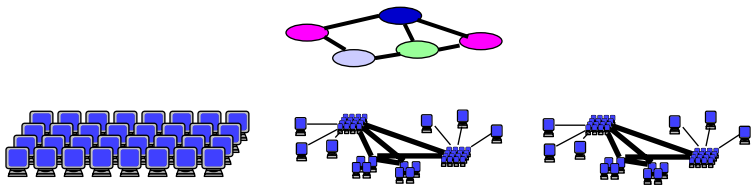


# Dynamic Grid Adaptation I



- Condor single-node view (asynchronous communication)
- Migration to other compatible resource or even a storage server (passivation)

# Dynamic Grid Adaptation II



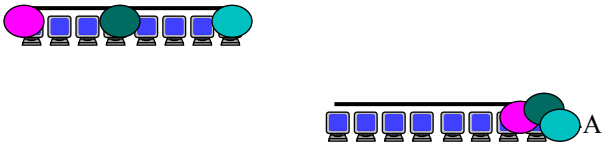
- Complex multiple process adaptation
  - » Change resources used
  - » Maintain application state
  - » Maintain/reestablish communication
  - » Change type of resource configuration used



# Intelligent Dynamic Adaptation

- Understand Application Performance
- Understand how it relates to Resources
  - » e.g. what happens if R-A is faster than expected, what if network-B is slower than expected, if we add three more x86 PC's, etc.
- Know current resource status: performance and availability
- Know about future resource status: performance and availability
  - » dynamic sharing, aggressive and malicious competition
- => Just from the application perspective (RM side comes later)

# Intelligent Control is not Simple




- T=0, all Apps see that R-A is available (high performance and low load)
- T=1, all Apps decide to migrate or incorporate to R-A
- T=2, all Apps using R-A see much lower performance than they anticipated
  - » All never payback the cost of migration
  - » Several see lower performance absolute performance than before the move



# Discussion

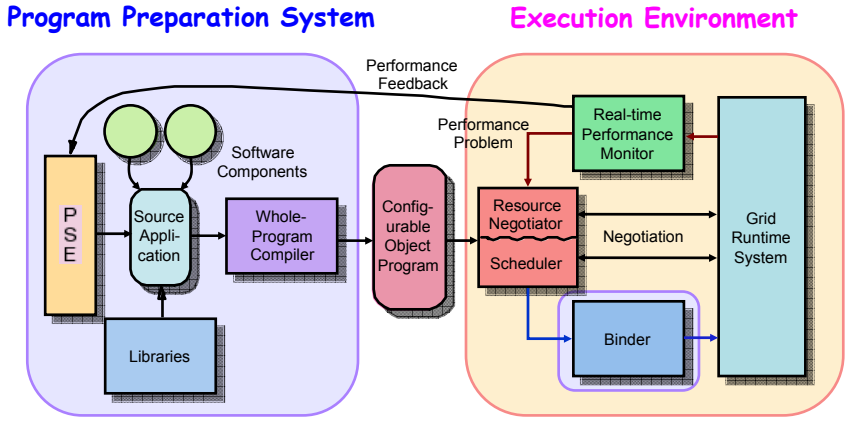
- What do we understand about application performance?
  - » Relation to resource configuration selected
  - » Relation to change in a single resource
  - » Adding one, removing one
- How much can we know about resource status? And future?
- On what other criteria might we want to adapt?

# Grid Application Development Software (GrADS) Project (1999-2003)

- Idea: Build Adaptive Application tools, so its not so hard for programmers
- Key elements of the architecture:
  - » Performance Model
  - » Initial Scheduler (selection; how many)
  - » Instrumented Application Program
  - » Distributed Resource Monitoring
  - » Decision Procedure for Resource Adaptation
  - » Rescheduling (selection again; how many)



# GrADSoft Architecture



# Application Performance Model

- Scalapack (matrix dimensions, number of processors, communication performance) => scalar
- FASTA (number of processors, sequence size) => scalar
- BLAST/PSI-Blast/EOL (sequence size, database size, ?) => scalar
- EMAN (tasks, iterations, ?) => scalar

- => these are all distributed, concurrent applications with communication amongst the parts



# Initial Scheduler

- Inputs:
  - » Application Performance Model
  - » Resource Universe (dump of the entire MDS system)
  - » Heuristic starting from largest chunk of resources (Local cluster, part of a big parallel machine)
    - – Add remote resources incrementally
    - – No consideration of load or status
- Outputs:
  - » Number of processors/resources to use
  - » Which resources to use
- This is a really hard problem: consider how many choices
- Comprehensive Resource Selection

# Detecting need for Change and Acting on it

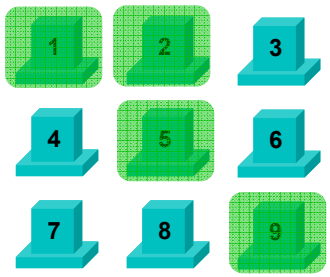
- Instrumented Application (contract monitor)
  - » Binary rewriting of application
  - » Addition of instrumentation based on structural analysis of program
  - » Understand control structure and communication operations
- Distributed Resource Monitoring
  - » Things like Network Weather Service
  - » Other information services (e.g. Ganglia)
- Decision Procedure for Resource Adaptation
  - » Didn't meet the "contract"



# Two Rescheduling Systems

- UCSD (Sievert, Casanova)
  - » Pre-allocate more than needed resources (N > n); monitor for those that might speed your execution
  - » Shift your n processes to the fastest of the N resources
  - » Shuffling the data within an MPI communicator
- UTK (Vadhiyar, Dongarra)
  - » Trigger reschedule; estimate right next set of resources
  - » Write checkpoint to IBP Depot
  - » Restore checkpoint on new resources

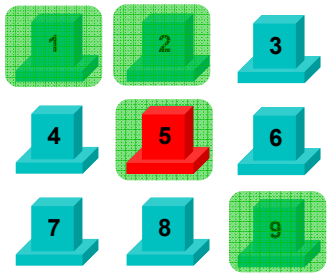
# N to N: Usage Scenario




- User Launches App on 4 Processors
- Processor 5 Slows Down Because of Sudden Heavy Interactive Use
- App Senses Slowdown, Moves Computation From Processor 5 to Processor 7

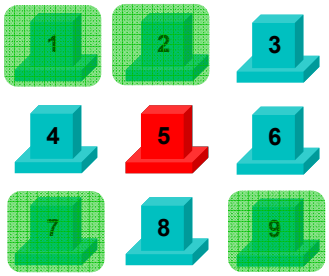


# Usage Scenario




- User Launches App on 4 Processors
- Processor 5 Slows Down Because of Sudden Heavy Interactive Use
- App Senses Slowdown, Moves Computation From Processor 5 to Processor 7

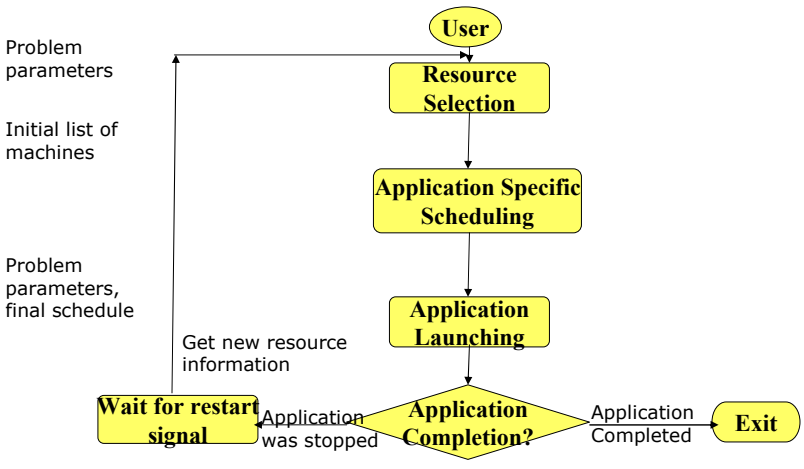
# Usage Scenario




- User Launches App on 4 Processors
- Processor 5 Slows Down Because of Sudden Heavy Interactive Use
- App Senses Slowdown, Moves Computation From Processor 5 to Processor 7
- => additional decision procedure based on estimate of load on newly selected machine



# Rescheduling GrADS – (M-N)

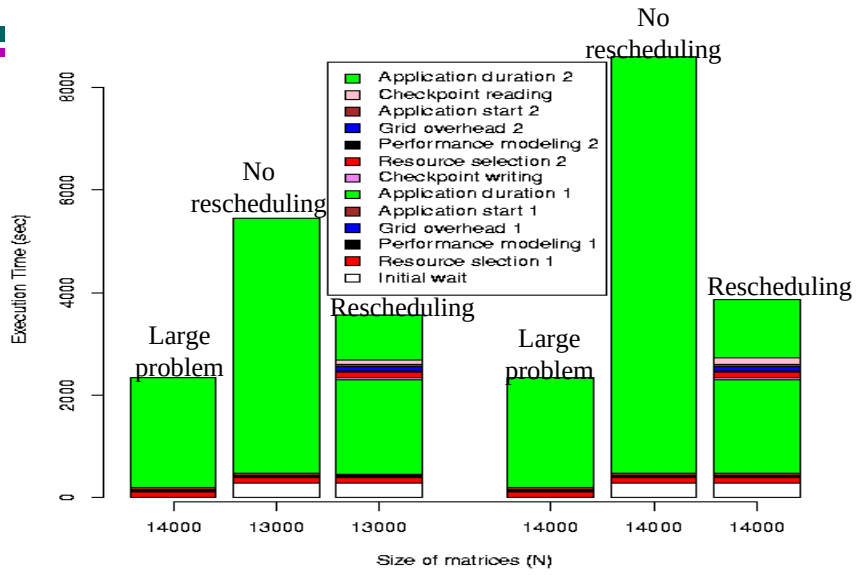


# Rescheduling Framework Components

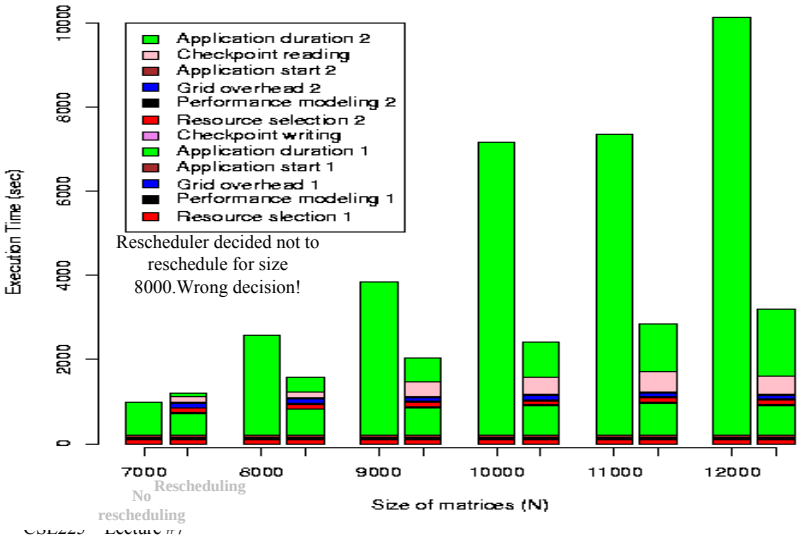
- Migrator
  - » User level checkpointing library (SRS)
- Contract Monitor
  - » Uses tolerance limits specified by app
  - » Adjusts tolerance limits if rescheduler decides not to migrate
- Rescheduler
  - » Evaluates benefits of rescheduling wrt remaining execution time
  - » Migration on request (via contract violation) or opportunistic (other apps complete)



# Experiments and Results



# Experiments and Results
## Rescheduling on request

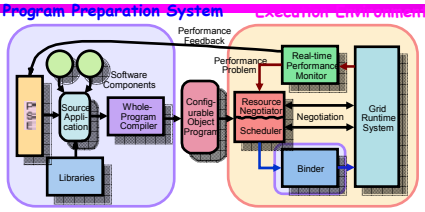





# VGrADS: Virtual Grid Application Development Software

- 5-year Follow-on Project, started Fall 2003
- Observations:
  - » Building performance models for applications is hard
  - » Scheduling subject to them is just as hard
  - » Deciding when to reschedule is hard
- VGrADS: Move to a Simpler Model
  - » Simple set of non-application-specific performance models
  - » Decouple application optimization from model realization (scheduling)
  - » Move to proactive performance monitoring and advance to detecting performance opportunities
  - » Enable scalable resource management and adaptation
  - » Enable consideration of other issues, such as Fault-tolerance

# Lessons from GrADS




- Approach Vindicated: Application Driven Adaptation is Crucial
  - » Doing this well is DIFFICULT
- Specifically:
  - » Implicit Coupling of Application, Programming Tools, and Runtime requires dealing with complexity at all of these levels simultaneously
  - » Lack of Explicit Resource Abstraction inhibits expressing and exploiting application domain knowledge for application and resource management
  - » Closed World Selection Model does not extend to larger, shared resource Grid environments with contended allocation



# Virtual Grid Research Challenges

- Separation of Concerns
  - » Application Planning and Management
  - » Complex Grid Resource Environment Mgmt
- Scalable Selection and Binding
  - » Large Resource Pools
  - » Competitive, Dynamic Environments
- Application-Driven Resource Management
  - » Abstraction Level
  - » Grid Information
  - » Support Fault-Tolerance and Reasoning about Behavior

# Virtual Grid Approach

- Separation of Concerns
  - » "Application Level" Resource Abstraction
  - » vgDL: Virtual Grid Description Language
  - » Virtual Grid
- Scalable Selection and Binding
  - » Integrated "Finding and Binding"
  - » Overselection and Dynamic Composition
- Application-Driven Resource Management
  - » VG: Explicit Application Resource Abstraction
  - » VG: Unified Resource Information Provider
  - » VG: Launch and Monitor Computations
  - » VG: Modify to Manage Application Resources



# Summary

- What is needed for Dynamic Adaptive Applications?
  - » Resource-aware Applications; understand their performance
- GrADS and Rescheduling Systems
  - » Adaptive Application Frameworks
  - » Performance Model, Selection, Monitoring, Decision, Migration, Reselection
- VGrADS Project
  - » Simpler Model
  - » Separation of Concerns