New Grid Scheduling and Rescheduling Methods

Within the GrADS Project*

***Sponsored by NSF NGS**

Ken Kennedy

Center for High Performance Software

Rice University

http://www.cs.rice.edu/~ken/Presentations/GrADS-IPDPS04.pdf

GrADS
Grid Application Development Software Project

Principal Investigators

Francine Berman, UCSD

Andrew Chien, UCSD

Keith Cooper, Rice

Jack Dongarra, Tennessee

Ian Foster, Chicago

Dennis Gannon, Indiana

Lennart Johnsson, Houston

Ken Kennedy, Rice

Carl Kesselman, USC ISI

John Mellor-Crummey, Rice

Dan Reed, UIUC

Linda Torczon, Rice

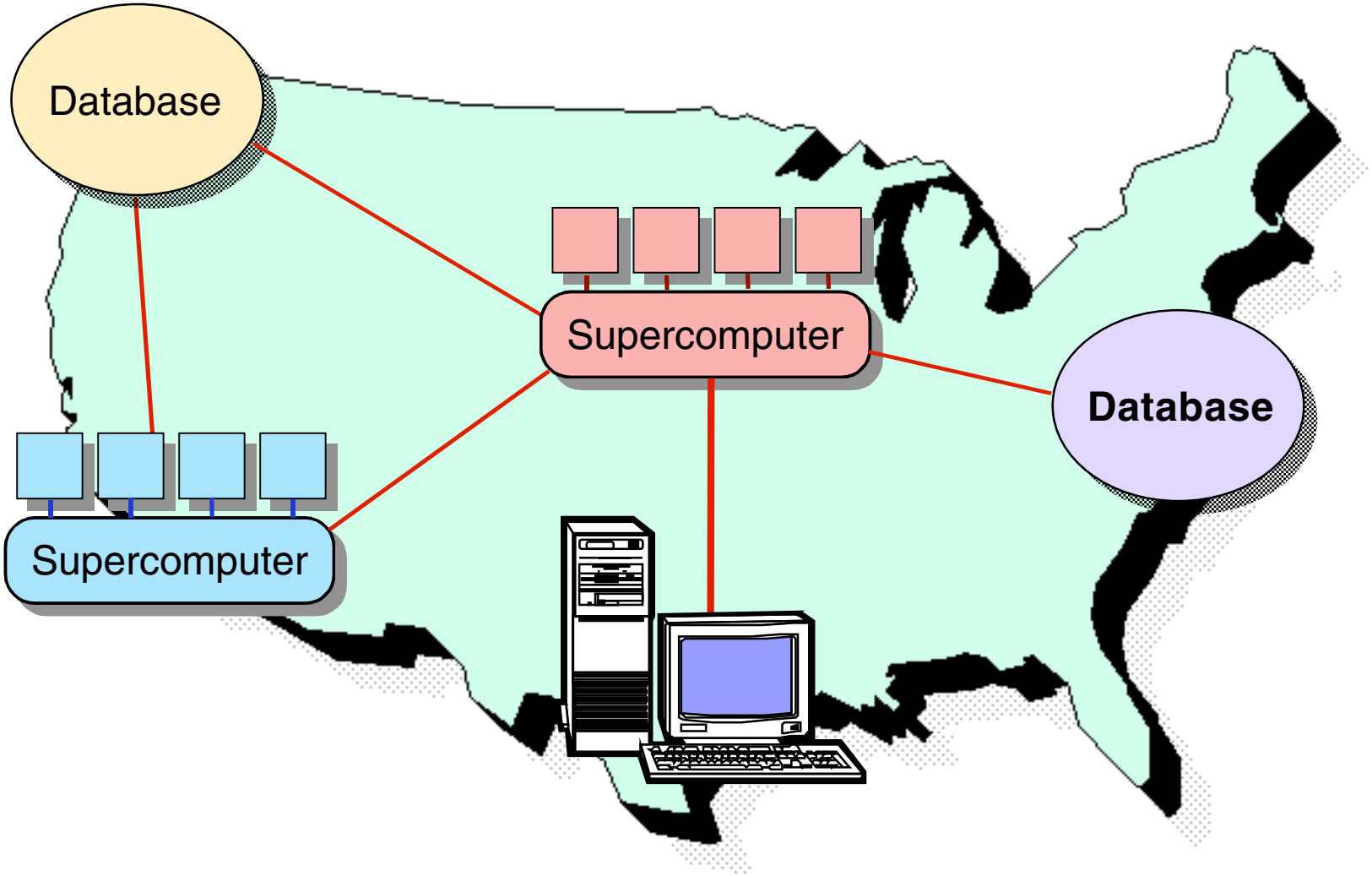
Rich Wolski, UCSB

Other Contributors

Dave Angulo, Chicago
Henri Casanova, UCSD
Wahid Chrabakh, UCSB
Holly Dail, UCSD
Anshu Dasgupta, Rice
Wei Deng, UIUC
Sridhar Gullapalli, USC ISI
Charles Koelbel, Rice
Bo Liu, UH
Xin Liu, UCSD
Anirban Mandal, Rice
Gabriel Marin, Rice

Mark Mazina, Rice
Celso Mendes, UIUC
Dragan Mirkovic, UH
Alex Olugbile, UCSD
Mitul Patel, UH
Zhiao Shi, Tennessee
Otto Sievert, UCSD
Martin Swany, Delaware
Sathish Vadhiyar, Tennessee
Shannon Whitmore, UIUC
Huaxia Xia, UCSD
Asim YarKhan, Tennessee

National Distributed Problem Solving



GrADS Vision

- **Build a National Problem-Solving System on the Grid**
  - Transparent to the user, who sees a problem-solving system
- **Software Support for Application Development on Grids**
  - Goal: Design and build programming systems for the Grid that broaden the community of users who can develop and run applications in this complex environment
- **Challenges:**
  - Presenting a high-level application development interface
    - If programming is hard, the Grid will not not reach its potential
  - Designing and constructing applications for adaptability
  - Late mapping of applications to Grid resources
  - Monitoring and control of performance
    - When should the application be interrupted and remapped?

Today: Globus

- Developed by Ian Foster and Carl Kesselman
  - Grew from the I-Way (SC-95)
- Basic Services for distributed computing
  - Resource discovery and information services
  - User authentication and access control
  - Job initiation
  - Communication services (Nexus and MPI)
- Applications are programmed by hand
  - Many applications
  - User responsible for resource mapping and all communication
    - Existing users acknowledge how hard this is

Today: Condor

- Support for matching application requirements to resources
  - User and resource provider write ClassAD specifications
  - System matches ClassADs for applications with ClassADs for resources
    - Selects the "best" match based on a user-specified priority
  - Can extend to Grid via Globus (Condor-G)
- What is missing?
  - User must handle application mapping tasks
  - No dynamic resource selection
  - No checkpoint/migration (resource re-selection)
  - Performance matching is straightforward
    - Priorities coded into ClassADs

GrADS Strategy

- Goal: Reduce work of preparing an application for Grid execution
  - Provide generic versions of key components currently built in to applications
    - E.g., scheduling, application launch, performance monitoring
- Key Issue: What is in the application and what is in the system?
  - GrADS: Application = configurable object program
    - Code, mapper, and performance modeler

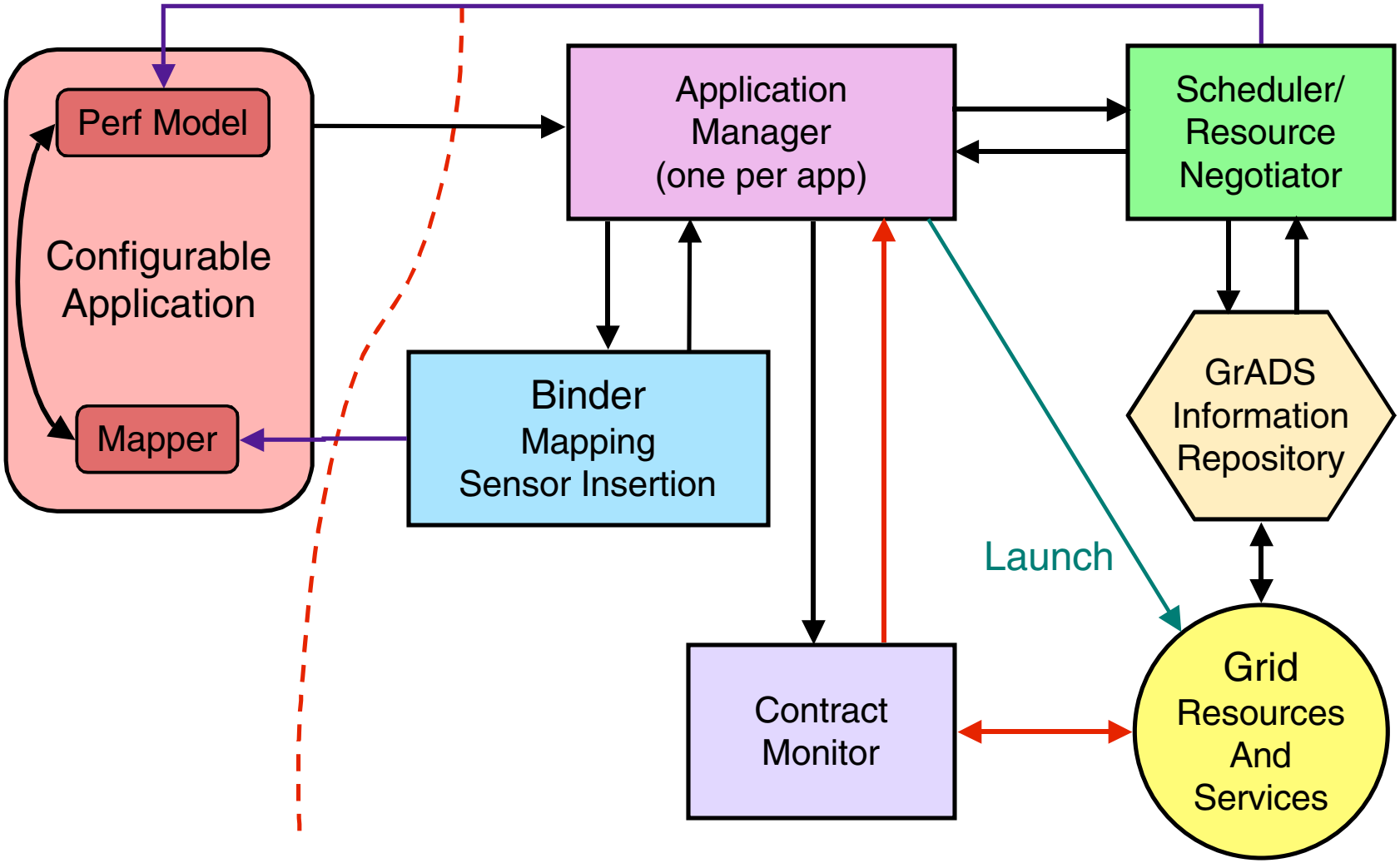
GrADS Strategy

- Approach: Define "Configurable Object Program" to describe application, separate out system concerns
  - Code
    - Today: DAGs of MPI programs; tomorrow: components & more general representations
  - Mapper
    - Given a set of resources, maps computation to those resources
  - Performance Model
    - Given a set of resources and mapping, estimates performance
    - Serves as objective function for system Resource Negotiator/Scheduler
- Automatically construct mappers and performance models

Configurable Object Program

- Goal: Provide minimum needed to automate resource selection and program launch
- Code
  - Today: MPI program
  - Tomorrow: more general representations
- Mapper
  - Defines required resources and affinities to specialized resources
  - Given a set of resources, maps computation to those resources
    - "Optimal" performance, given all requirements met
- Performance Model
  - Given a set of resources and mapping, estimates performance
  - Serves as objective function for Resource Negotiator/Scheduler

GrADS Program Execution System



New GrADS Work for 2003-2004

- **New Binder**
  - Supports heterogeneous execution, preinstalled components
- **New Workflow Scheduler**
  - Supports DAG scheduling using performance models
- **Rescheduling**
  - N-N rescheduling based on process migration
  - N-M rescheduling based on checkpoint and restart
- **Experiments**
  - New applications
    - EMAN — 3D image reconstruction
    - Particle-in-cell code for N-N experiments
  - MicroGrid used for rescheduling experiments

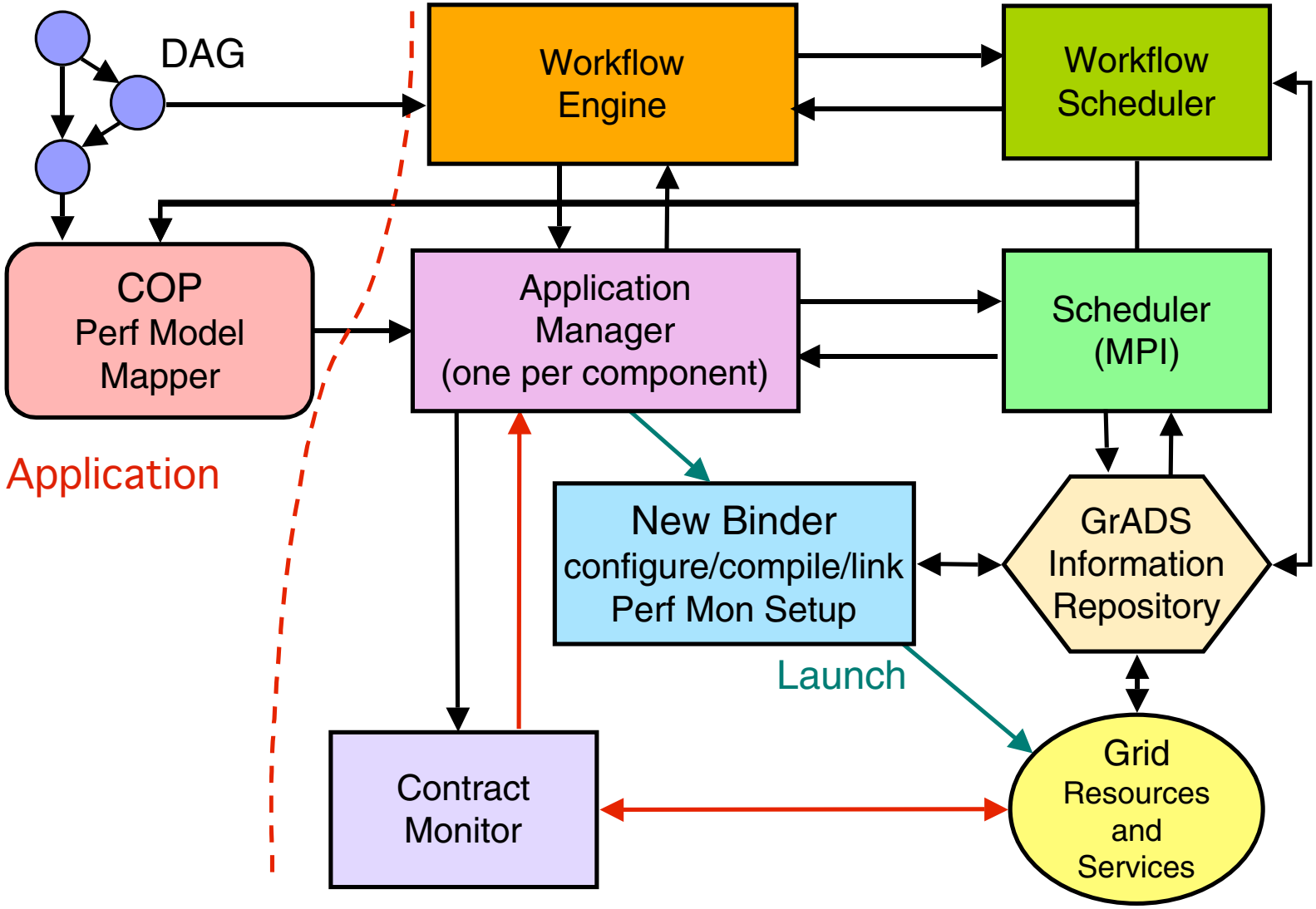
New Binder

- Software libraries preinstalled on various resources on Grid
  - Catalogued in GrADS Information System
  - Performance models included
- Launcher stages source files along with a script to each of the target machines
  - Script includes sensor insertion, compilation of staged components, and linking against preinstalled libraries
  - If component = MPI job
    - control is passed back to the application manager and launch proceeds after synchronization
  - If component ≠ MPI job
    - execution proceeds immediately, with notification to the application manager at the end

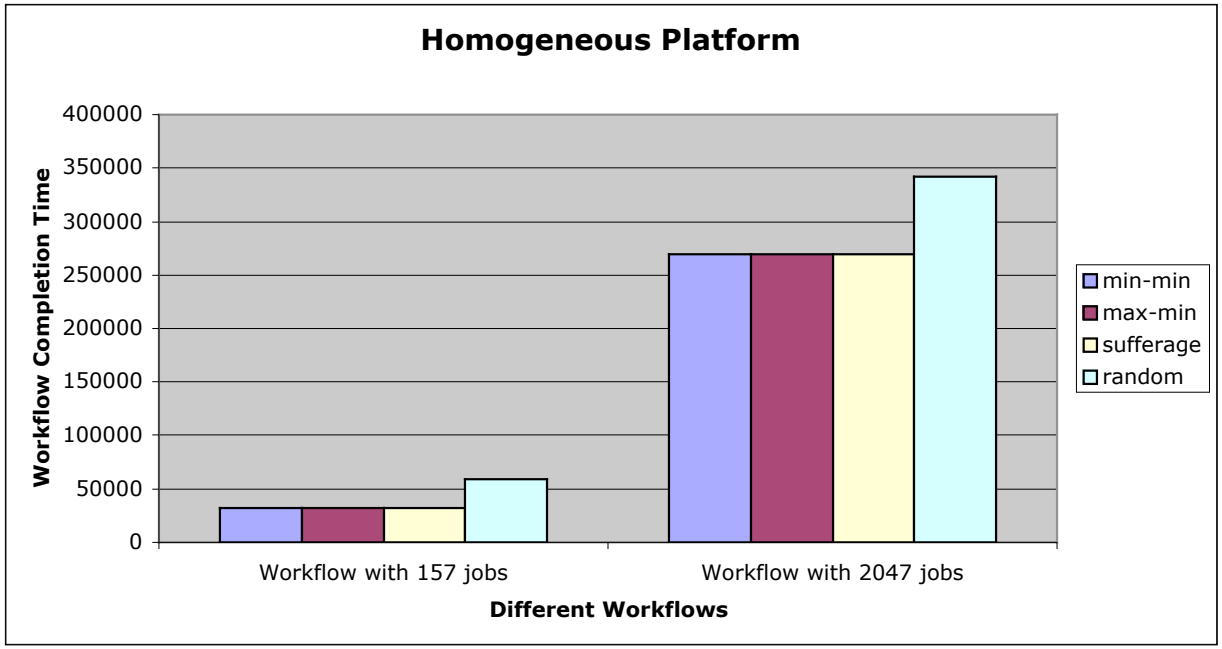
Workflow Scheduling

- Many Grid applications can be represented as workflow directed acyclic graphs
  - Currently these are scheduled using an on-demand scheduler such as Condor DAGMan
  - When data movement costs are taken into account, poor schedules can result
    - Computations stuck at suboptimal resources
- GrADS performance models provide opportunity to do global DAG scheduling
  - Matches most intensive computations to the best resources over the entire DAG, rather than when tasks become ready
  - Significant performance advantages obtained

GrADS:Workflow and New Binder/Launcher



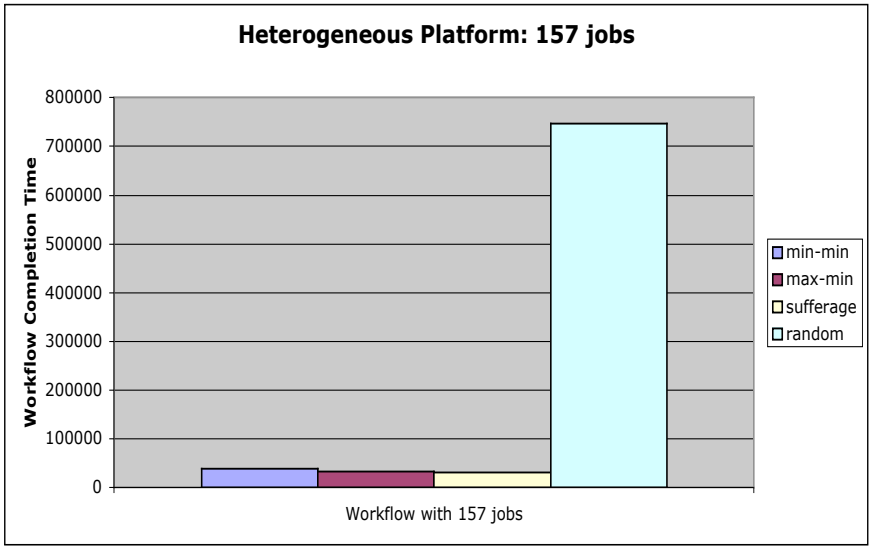
Heuristic Workflow Scheduling: Results

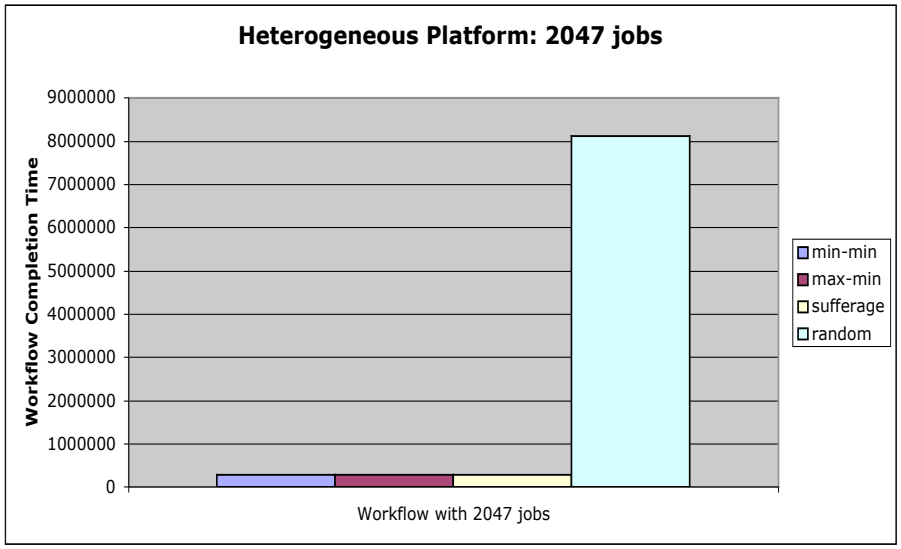


- **Simulation results for workflow completion times for different "Montage" workflows**
- **Improvement of >20% for homogeneous platform**

Preliminary results from joint work with Ewa Deelman *et. al.* at USC ISI

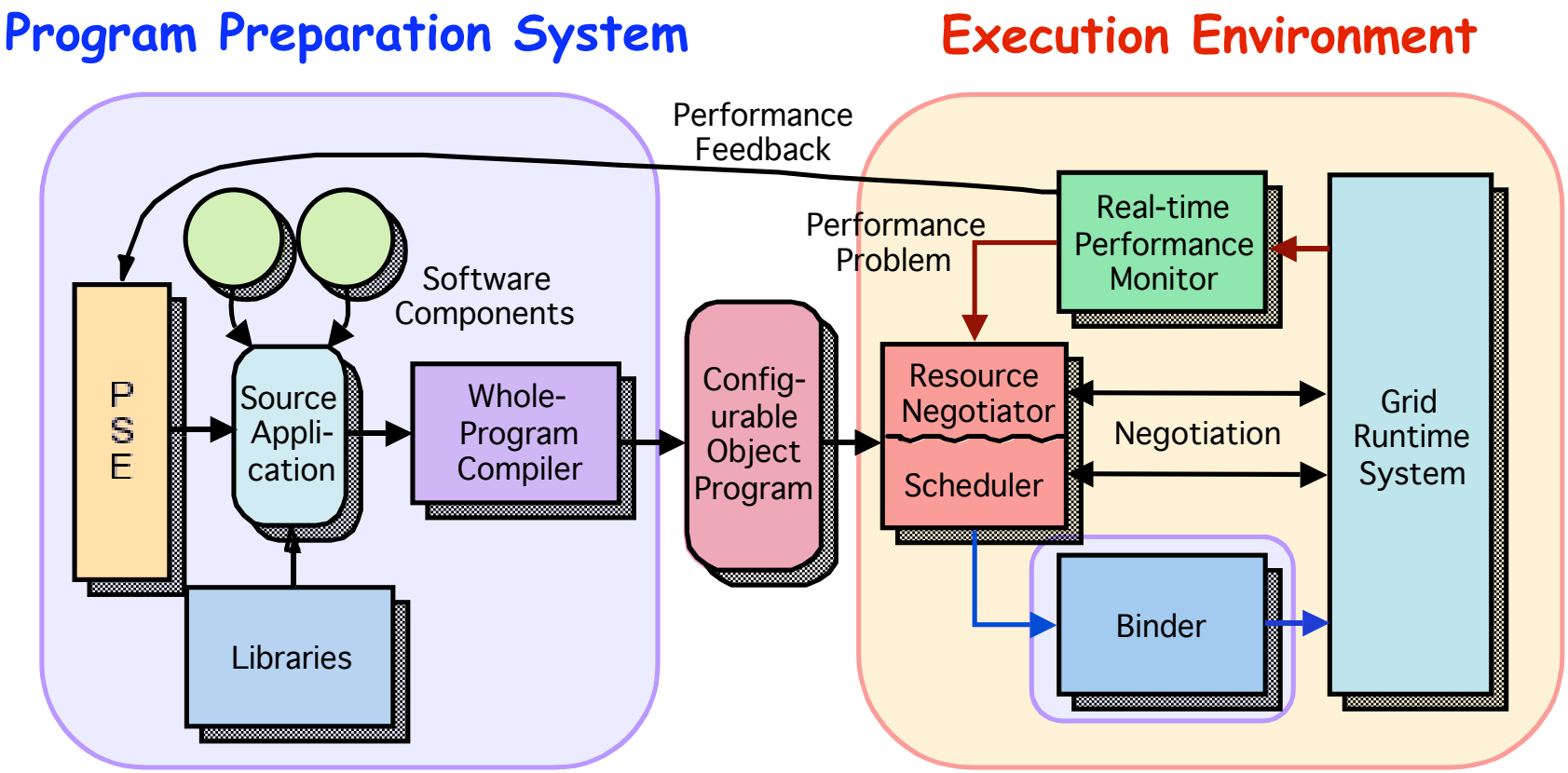
Heuristic Workflow Scheduling: Results



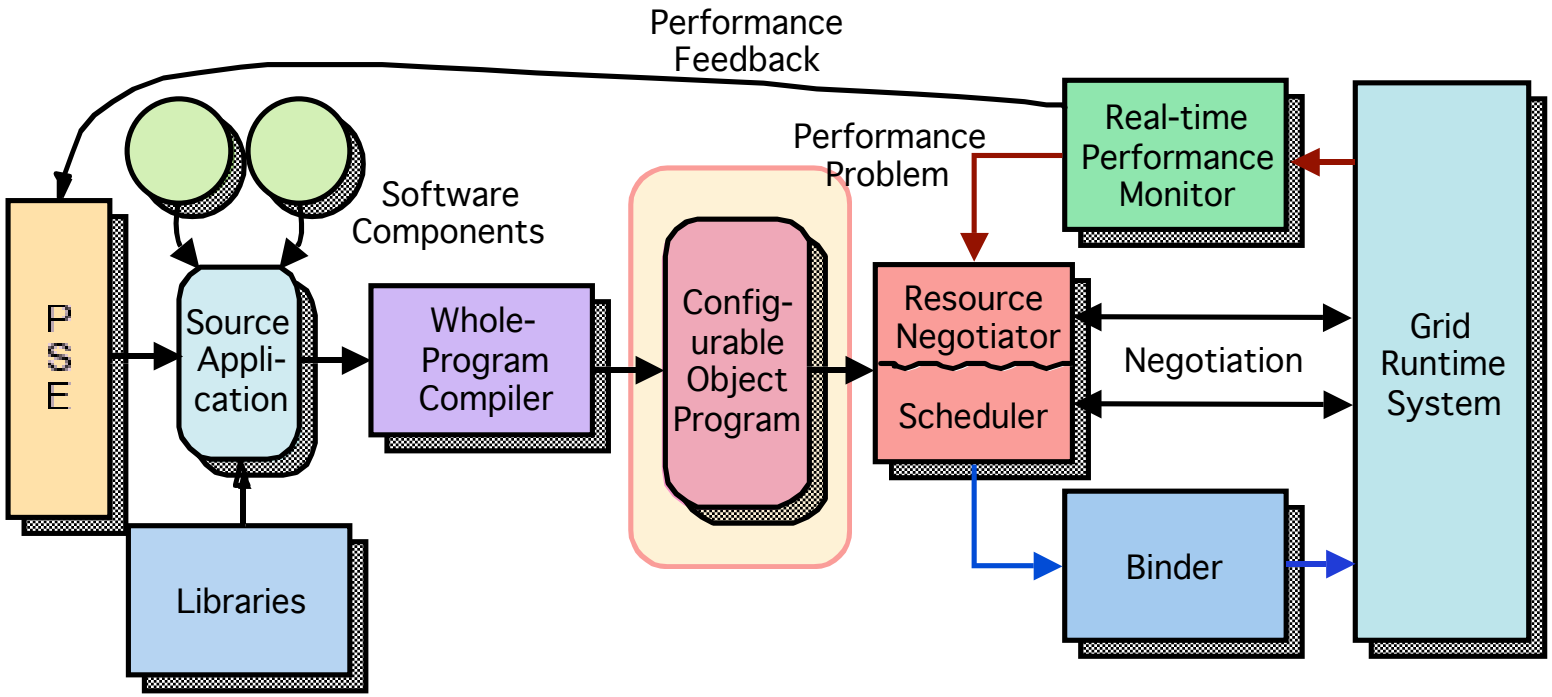


- **By using heuristic workflow scheduling, workflow completion times improve by an order of magnitude[>20 times] over random scheduling for heterogeneous platform**
- **Workflow completion time is within 10% of that using a very expensive AI scheduler that doesn't scale to 2047 jobs**

GrADSoft Architecture

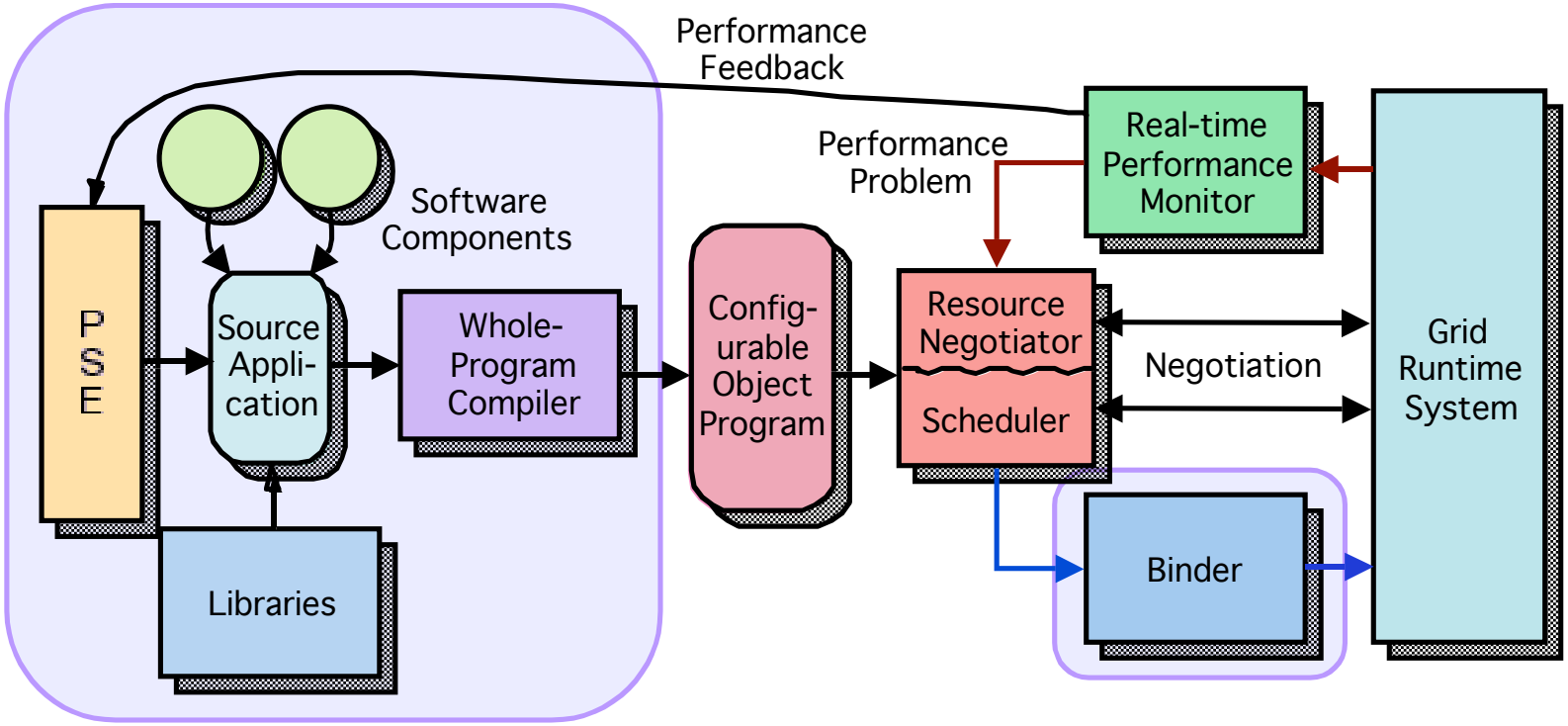


GrADSoft Architecture



GrADSoft Architecture

Program Preparation System



Program Preparation Tools

- Goal: provide tools to support the construction of Grid-ready applications (in the GrADS framework)
- Performance modeling
  - Challenge: synthesis and integration of performance models
    - Combine expert knowledge, trial execution, and scaled projections
  - Focus on binary analysis, derivation of scaling factors
- Mapping
  - Construction of mappers from parallel programs
    - Mapping of task graphs to resources (graph clustering)
  - Integration of mappers and performance modelers from components
- High-level programming interfaces
  - Problem-solving systems: integration of components

Program Preparation System

- Goal: provide tools to support the construction of Grid-ready applications (in the GrADS framework)
- Performance models
  - Challenge: synthesis and integration of performance models
  - Approaches: exploit application knowledge, analysis of binaries, test runs, derivation of scaling factors
- Mappers
  - Challenges: Construction of mappers from MPI programs and workflow graphs
  - Approach: Mapping of task graphs to resources (graph clustering)
- Improved Binder
  - Challenges: Heterogeneous platforms, instrumentation insertion
  - Approach: Stage source code rather than binaries

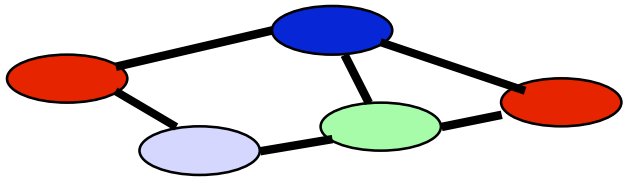
Execution System

- Goal: Provide efficient execution of programs in dynamic Grid environments
- Scheduling Workflow Applications
  - Challenge: Optimally mapping DAGs of heterogeneous processes
  - Approach: Heuristics based on performance models
- Rescheduling MPI Applications
  - Challenge: Load balancing tightly-coupled tasks without major application rewrite
  - Approach 1: overallocate processors, monitor load, migrate tasks from loaded to unloaded processors, hijack MPI calls for message re-routing
  - Approach 2: Detect imbalance, checkpoint computation, restart
- Information Service
  - Challenge: Manage information about state of the Grid
  - Approach: Network Weather Service, MDS

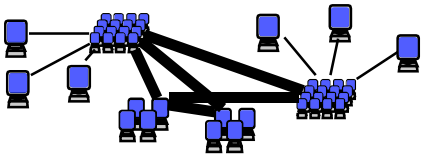
Testbeds

- **Goal:**
  - Provide vehicle for experimentation with the dynamic components of the GrADS software framework
- **MacroGrid**
  - Collection of processors running Globus and GrADS framework
  - At all GrADS sites (significant clusters at UCSD, UH, UIUC, UTK)
    - IA-32, IA-64, Sparc machines
  - Permits experimentation with real applications
- **MicroGrid**
  - Cluster of processors emulating the Grid
  - Runs standard Grid software (Globus, Nexus, GrADS middleware)
  - Permits simulation of varying loads and configurations
    - Controlled experiments
    - Stress GrADS components

MicroGrid Enables Deep Study of Grid Dynamics

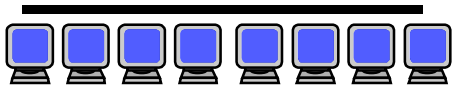


Grid Application

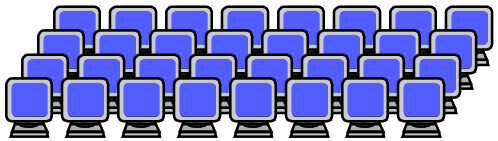


Virtual Grid, “MicroGrid”

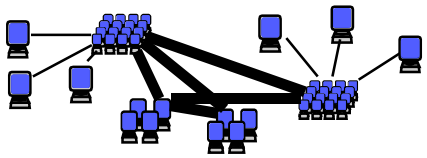
MicroGrid Software



LAN Workgroup



Scalable Cluster

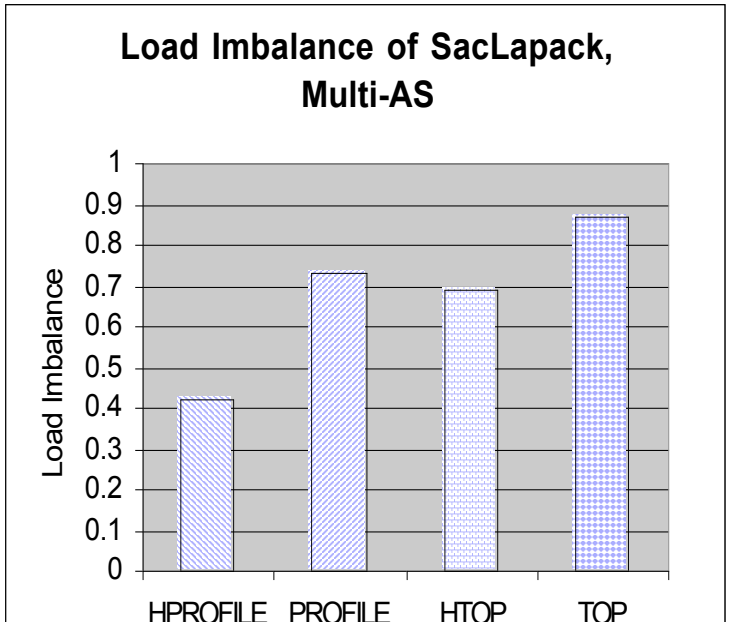


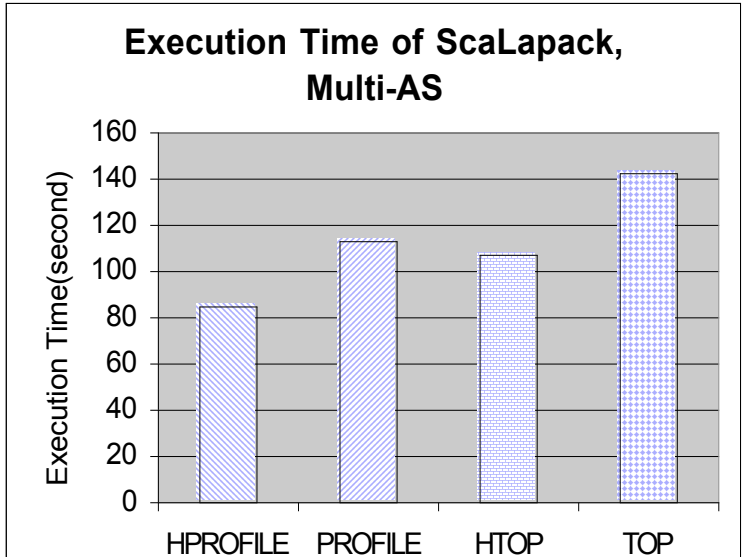
Heterogeneous Environment

MicroGrid Highlights

- Binary Interception enables Transparent Virtualization (SC2000)
- "Virtual time" enables wide range of relative performance experiments
- Scalable Packet-level Simulation provides accurate protocol behavior (SC2003)
  - Sophisticated Graph Partitioners using Topology and Profile Behavior
  - Full TCP, Router, OSPF, BGP, etc. modelling
- MicroGrid Simulator validated on wide range of benchmarks and grid applications (JOGC 2004)
- Released as Open Source (2/03, 7/03, Version 2.4.4. Feb 2004)
  - See http://www-csag.ucsd.edu/

Update: Large-Scale Simulations



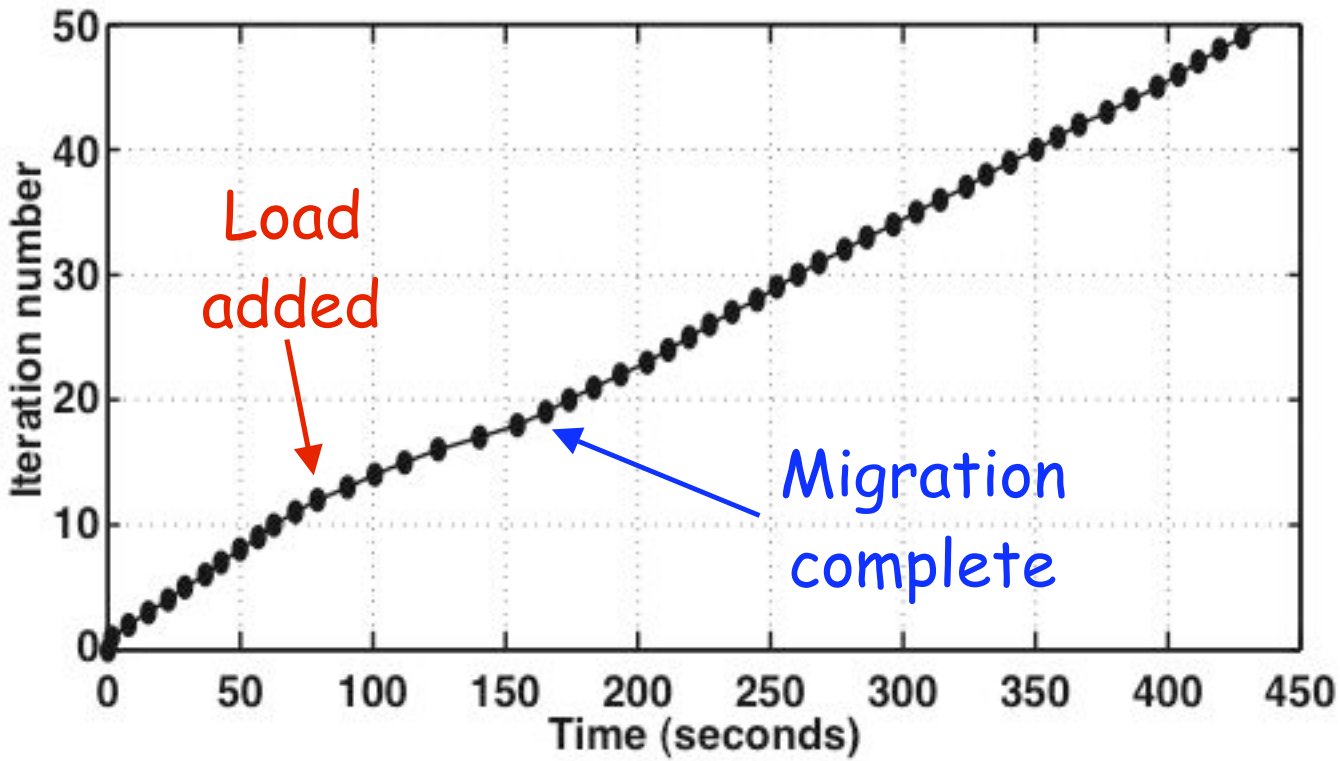


- **Scalable Grid Simulations Using the Clusters**
  - 128-node Itanium Cluster Simulation Engines (Teragrids)
  - Large-scale Network Structure Generation and Load Balancing Experiments
    - Flat networks of 20,000 routers (OSPF) (Scalable)
    - Hierarchical networks of 100 AS's with 200 routers each (BGP & OSPF) (Never done before!)

Rescheduling

- General Issues
  - Prediction of value of rescheduling
    - Cost of reschedule + performance on new resources
    - Estimate cost of current resources (unloaded)
- N-N Rescheduling
  - Allocate extra processors
  - Migration triggered by performance problem
  - Hijack MPI calls for message re-routing

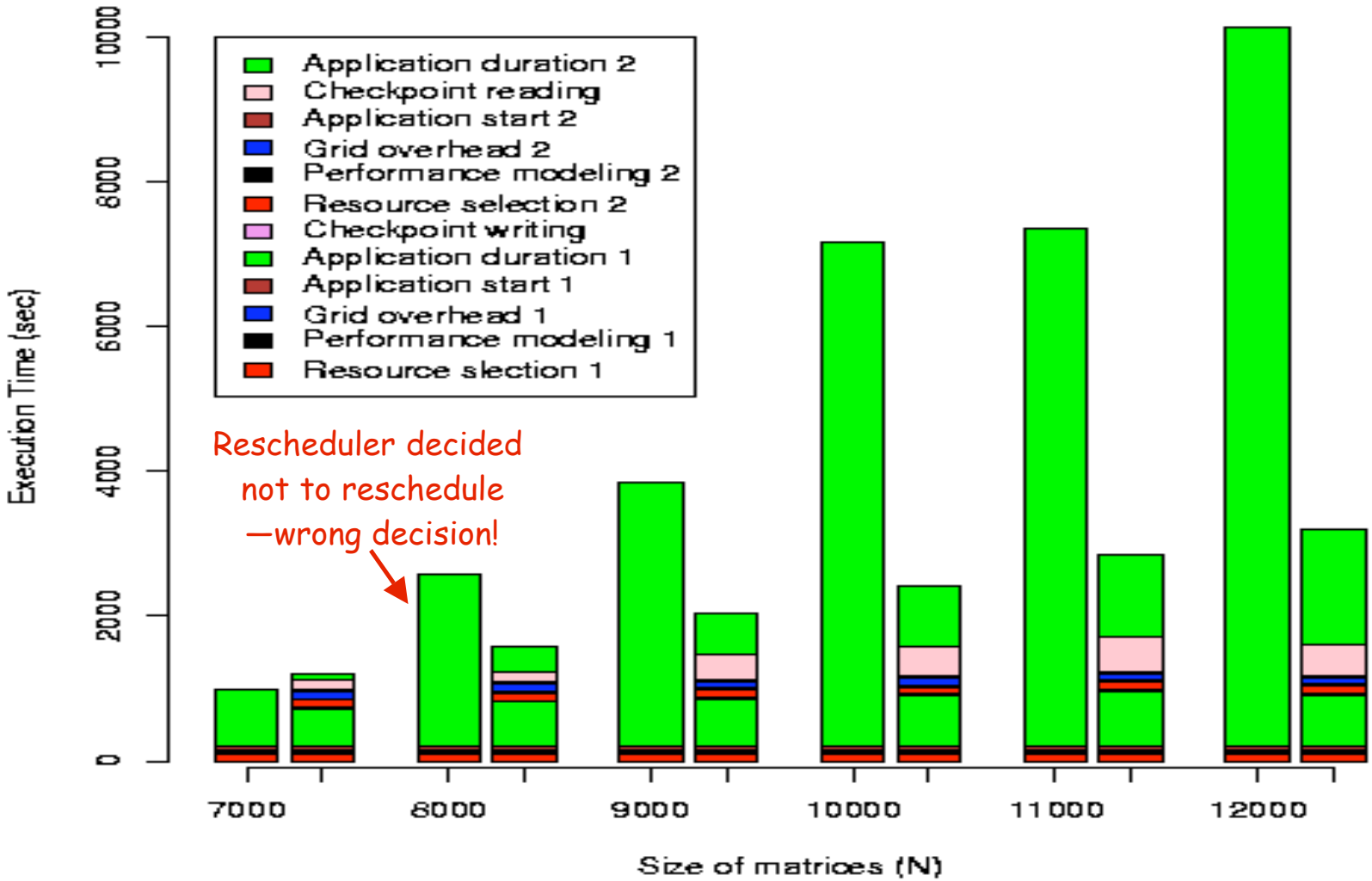
N-N Rescheduling



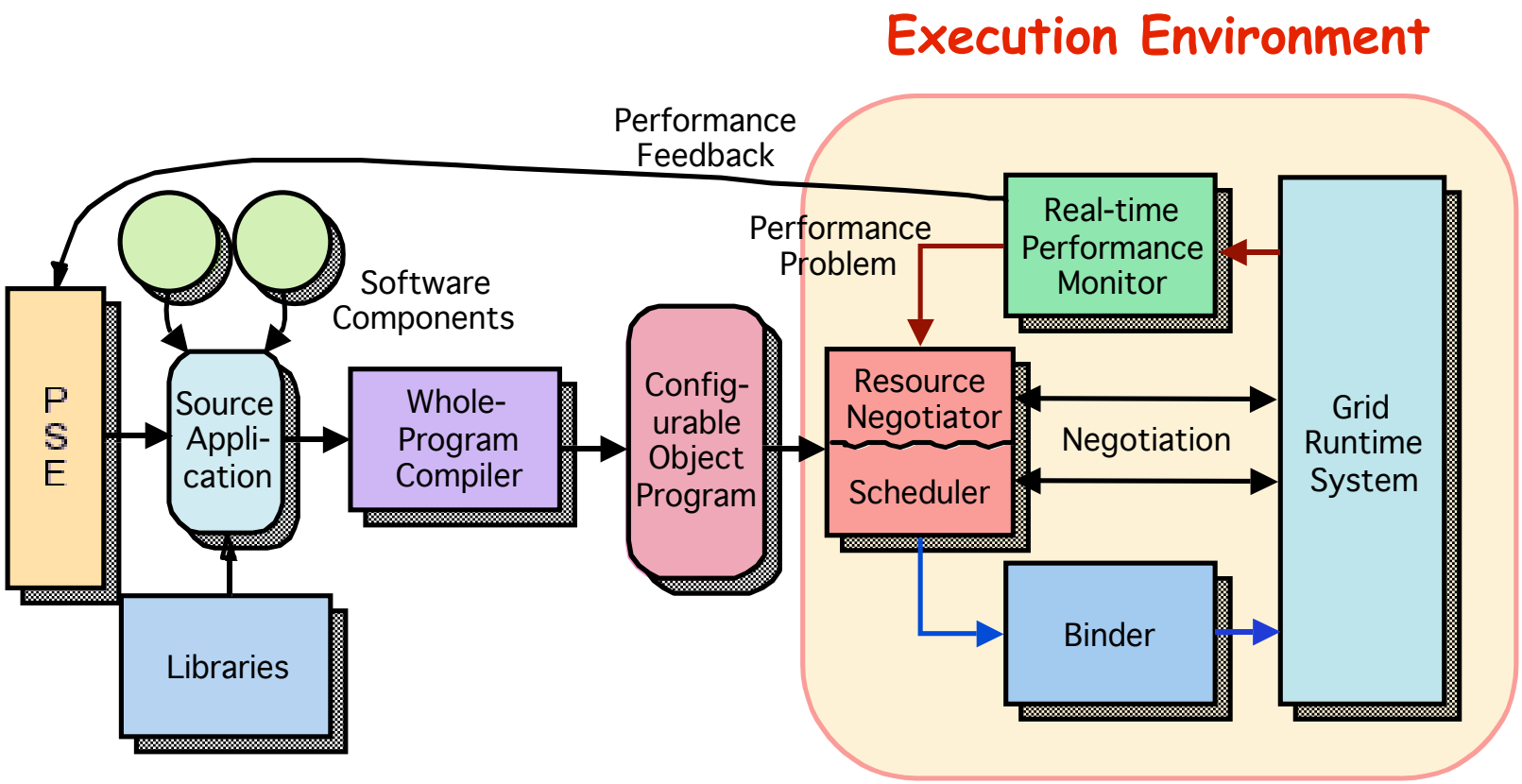
Rescheduling

- General Issues
  - Prediction of value of rescheduling
    - Cost of reschedule + performance on new resources
    - Estimate cost of current resources (unloaded)
- N-N Rescheduling
  - Allocate extra processors
  - Migration triggered by performance problem
  - Hijack MPI calls for message re-routing
- N-M Rescheduling
  - Checkpoint computation using library
  - Determine whether different resources would improve performance
  - Restart computation on new set of resources (or leave on old)

Performance of N-M Rescheduling



GrADSoft Architecture



Applications

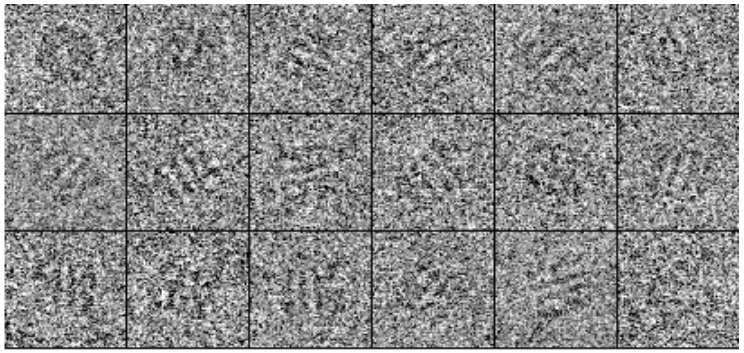
- N-body simulation
  - Particle-particle interaction by direct integration
  - N-to-N rescheduling: migrating work to unloaded processors
- ScaLAPACK
  - Linear algebra solver
  - N-to-M rescheduling: changing number of processors
- GridSAT
  - Boolean satisfiability: an NP-complete problem useful in circuit design and verification
- EMAN
  - Bio-imaging: 3D reconstruction of viruses from electron micrographs
  - Workflow scheduling: utilizing performance prediction to optimize mapping

Applications

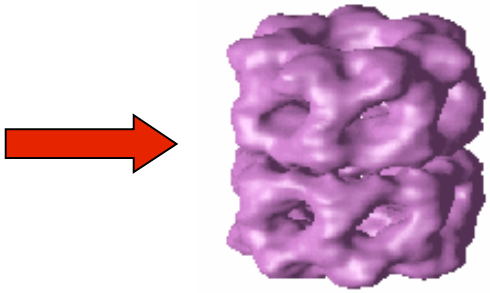
- N-body simulation
  - Particle-particle interaction by direct integration
  - N-to-N rescheduling: migrating work to unloaded processors
- ScaLAPACK
  - Linear algebra solver
  - N-to-M rescheduling: changing number of processors
- EMAN
  - Bio-imaging: 3D reconstruction of viruses from electron micrographs
  - Workflow scheduling: utilizing performance prediction to optimize mapping
  - Automatic performance model construction for both IA-32 and IA-64 clusters

A Workflow Application: EMAN

- **Applied Workflow Scheduling techniques to a Bio-Imaging application called EMAN**
- **Deals with 3D reconstruction of single particles from electron micrographs**

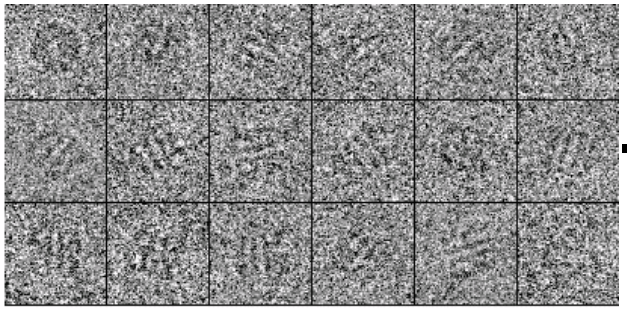


Electron Micrographs

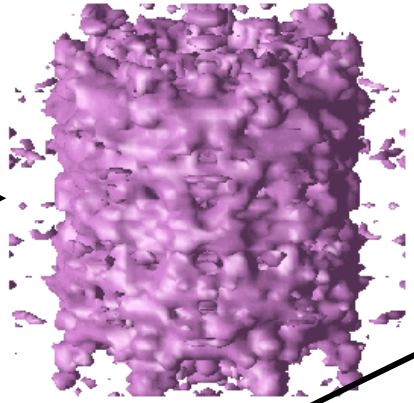


3D Model

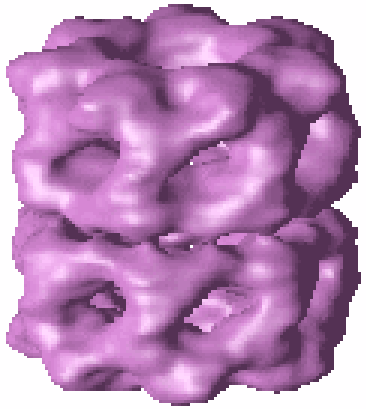
EMAN Refinement



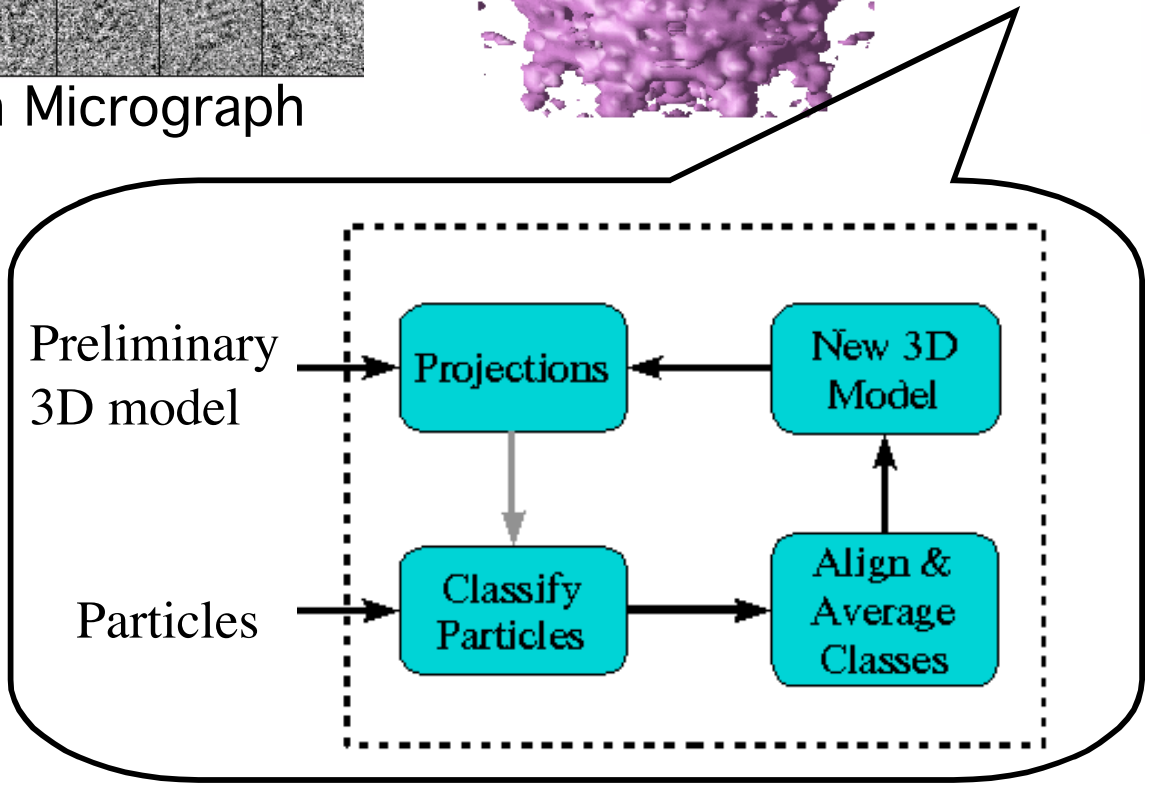
Electron Micrograph



Refine

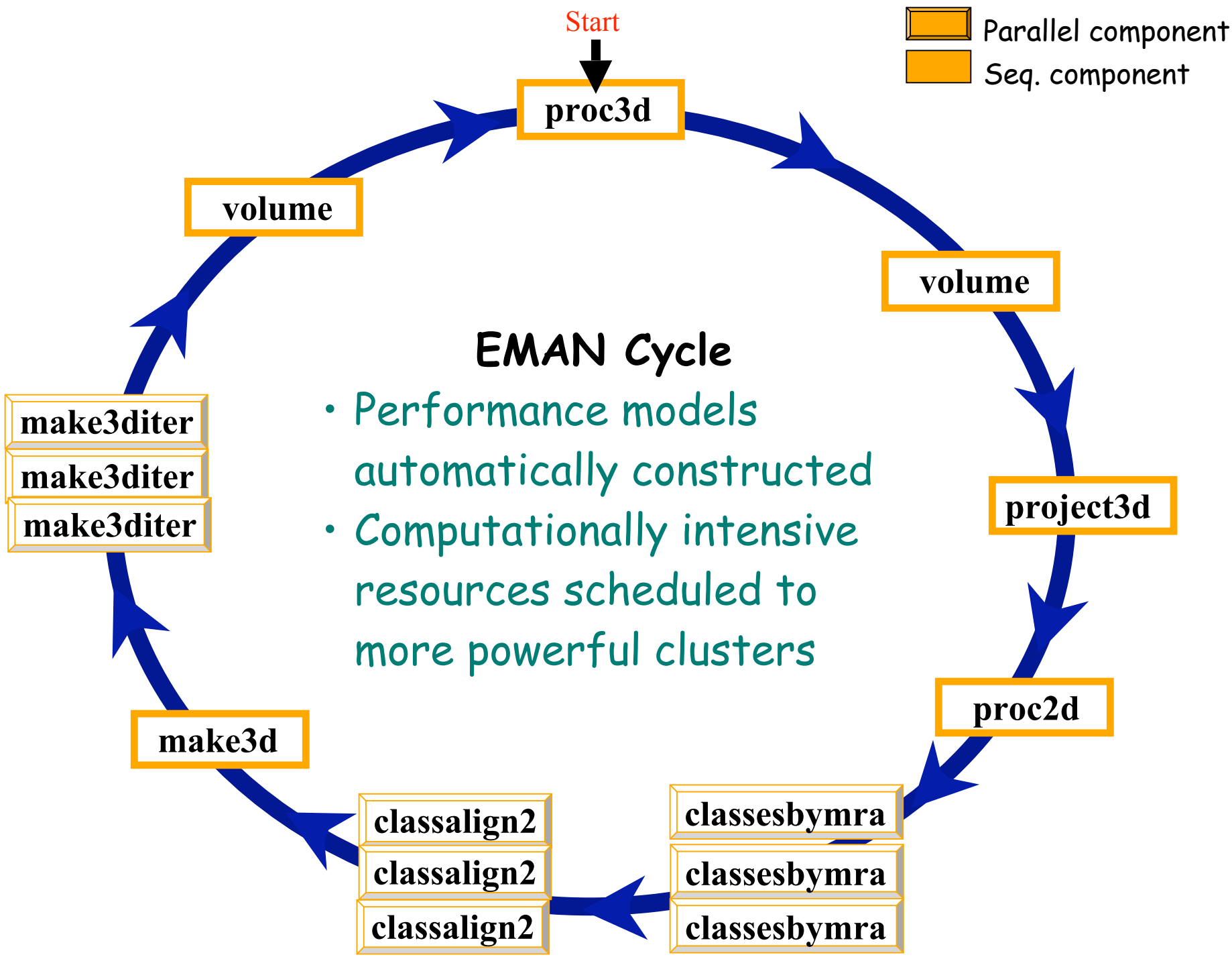


Final 3D model

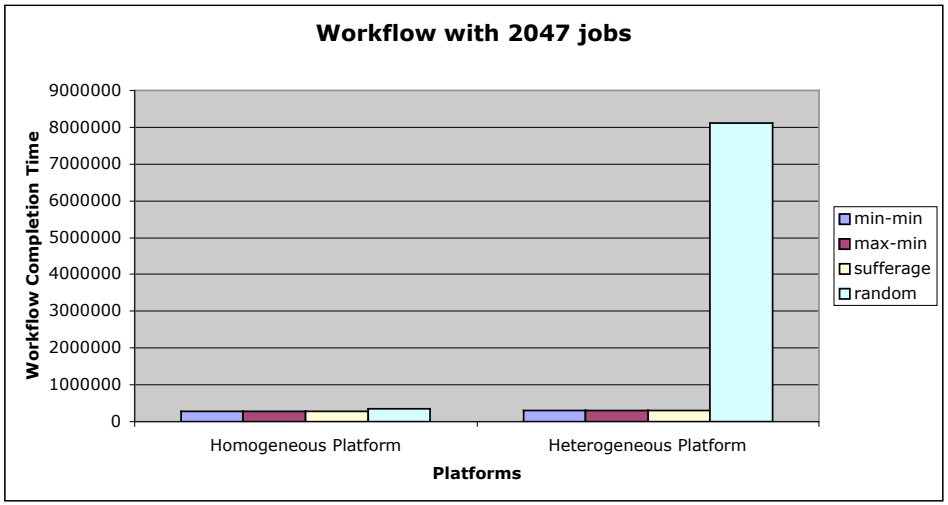


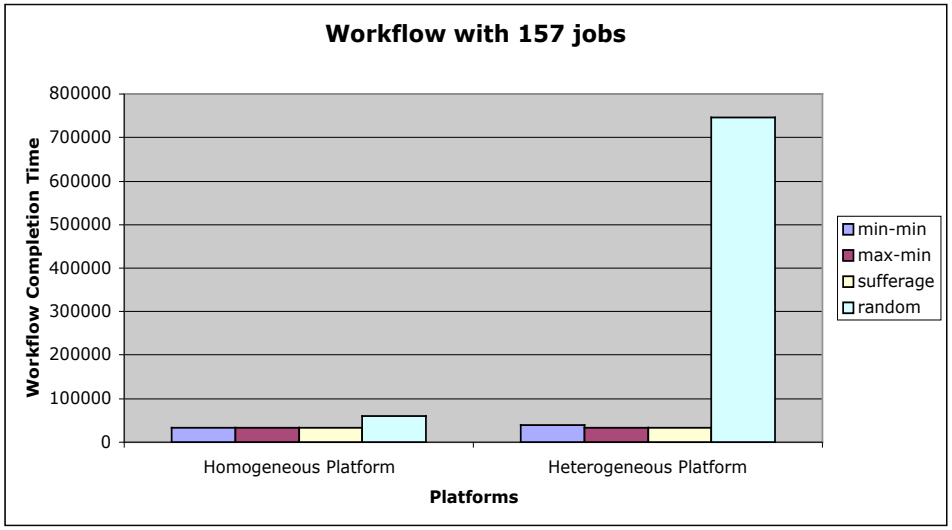
EMAN has been developed at Baylor College of Medicine by
Research group of ***Wah Chiu*** and ***Steven Ludtke*** {wah,sludtke}@bcm.tmc.edu

GrADS
Grid Application Development Software Project


Start
Parallel component
Seq. component
proc3d
volume
project3d
proc2d
classesbymra
classesbymra
classesbymra
classalign2
classalign2
classalign2
make3d
make3diter
make3diter
make3diter
volume
EMAN Cycle
• Performance models automatically constructed
• Computationally intensive resources scheduled to more powerful clusters

Heuristic Workflow Scheduling: Results



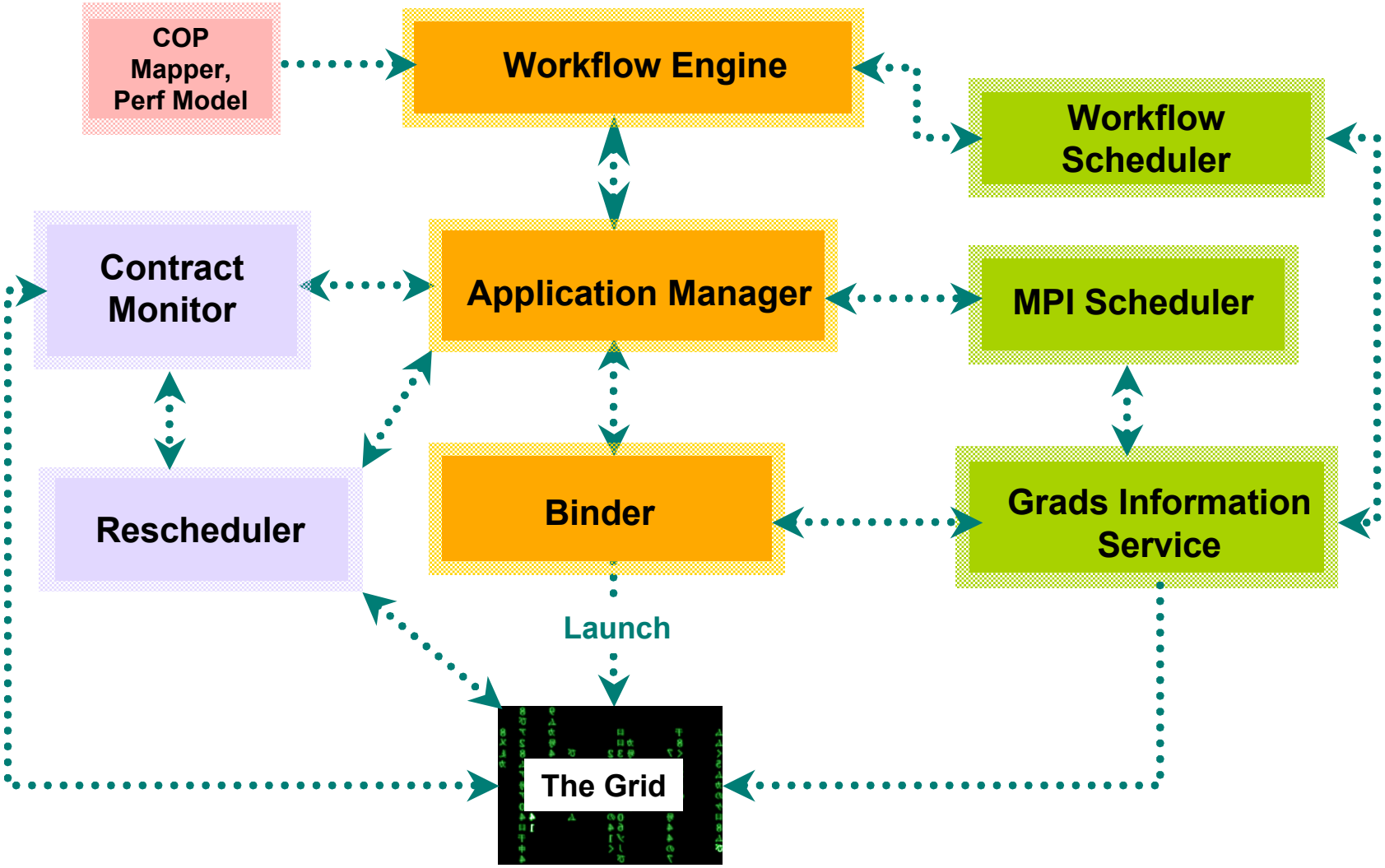


- Simulation results for Workflow completion times for different "Montage" workflows
- By using heuristic workflow scheduling, workflow completion times improve by an order of magnitude[>20 times] over random scheduling for heterogeneous platform
- Improvement of >20% for homogeneous platform
- Workflow completion time is within 10% of that using a very expensive AI scheduler that doesn't scale to 2047 jobs

Execution Cycle

- Configurable Object Program is presented
  - Space of feasible resources must be defined
  - Mapping strategy and performance model provided
- Resource Negotiator solicits acceptable resource collections
  - Performance model is used to evaluate each
  - Best match is selected and contracted for
- Execution begins
  - Binder tailors program to resources
    - Carries out final mapping according to mapping strategy
    - Inserts sensors and actuators for performance monitoring
- Contract monitoring is performed continuously during execution
  - Soft violation detection based on fuzzy logic

GrADS Program Execution Cycle



Demo Applications

- ScaLAPACK
  - LU decomposition of large matrices
- Cactus
  - Solver for gravitational wave equations
  - Collaboration with Ed Seidel's GridLAB
- FASTA
  - Biological sequence matching on distributed databases
- Smith-Waterman
  - Another sequence matching application using a strong algorithm
- Tomcatv
  - Vectorized mesh generation written in HPF
- Satisfiability
  - An NP-complete problem useful in circuit design and verification

Generation of Mappers

- Start from parallel program
  - Typically written using a communication library (e.g. MPI)
  - Can be composed from library components
- Construct a task graph
  - Vertices represent tasks
  - Edges represent data sharing
    - Read-read: undirected edges
    - Read-write in any order: directed edges (dependences)
    - Weights represent volume of communication
  - Identify oportunities for pipelining
- Use a clustering algorithm to match tasks to resources
  - One option: global weighted fusion

Testbeds

- Goal:
  - Provide vehicle for experimentation with the dynamic components of the GrADS software framework
- MacroGrid (Carl Kesselman)
  - Collection of processors running Globus and GrADS framework
    - Consistent software environment
  - At all 9 GrADS sites (but 3 are used for most experiments)
    - Availability listed on web page
  - Permits experimentation with real applications
- MicroGrid (Andrew Chien)
  - Cluster of processors (currently Compaq Alphas and x86 clusters)
  - Runs standard Grid software (Globus, Nexus, GrADS middleware)
  - Permits simulation of varying loads and configurations
    - Stress GrADS components (Performance modeling and control)

Summary

- **GrADS Project Goal:**
  - Design and build programming systems for the Grid that broaden the community of users who can develop and run applications in this complex environment
- **Strategy:**
  - Build an execution environment that automates the most difficult tasks
    - Map applications to available resources based on performance models
    - Transparently manage adapting to varying loads and changing resources
  - Automate the process of producing Grid-ready programs
    - Generate performance models and mapping strategies automatically, if possible
  - Construct programs using high-level domain-specific programming interfaces

Summary

- **Goal:**
  - Design and build programming systems for the Grid that broaden the community of users who can develop and run applications in this complex environment
- **Strategy:**
  - Build an execution environment that automates the most difficult tasks
    - Maps applications to available resources
    - Manages adapting to varying loads and changing resources
  - Automate the process of producing Grid-ready programs
    - generate performance models and mapping strategies semi-automatically
  - Construct programs using high-level domain-specific programming interfaces

VGrADS: Virtual Grid Application Development Software

- **Follow-on to GrADS**
  - NSF ITR award started October 1, 2003
- **New topics**
  - Programming Tools
    - Abstract Parallel Machine
    - Abstract Component Machine

      Like Grid Services or NetSolve or CCA or …
    - Nitty Gritty:

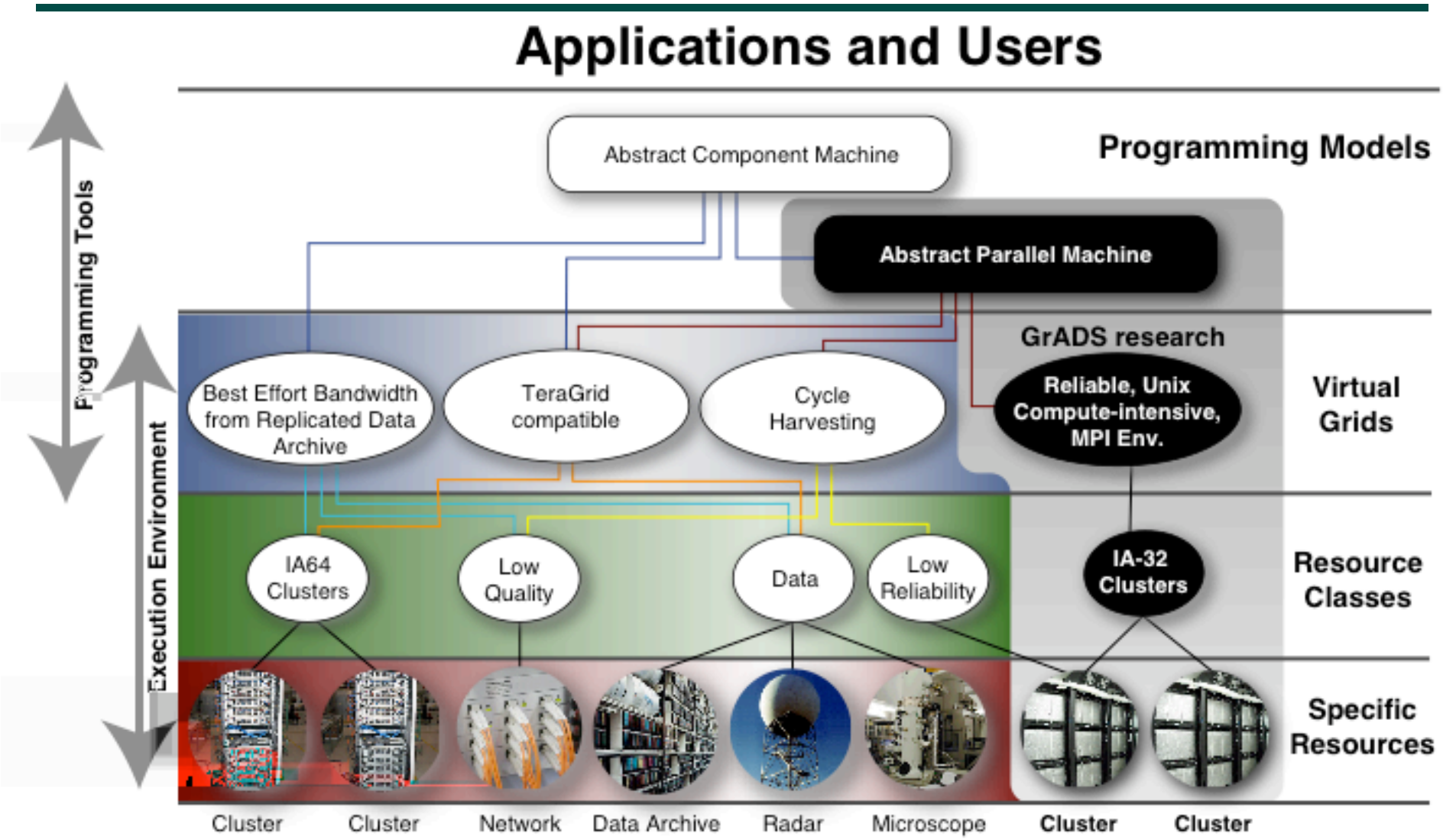
      Binder, Launcher, libraries, performance model generation, …
  - Execution System
    - Vgrids - Virtual Grids

      Provide structure & scalability via resource abstractions
    - Resource Abstraction Classes

      Resources in same class treated as identical for scheduling, etc.
    - Services to applications / programming tools / vgrids

      Virtualized scheduling and resource selection

VGrADS Vision



GrADS
Grid Application Development Software Project