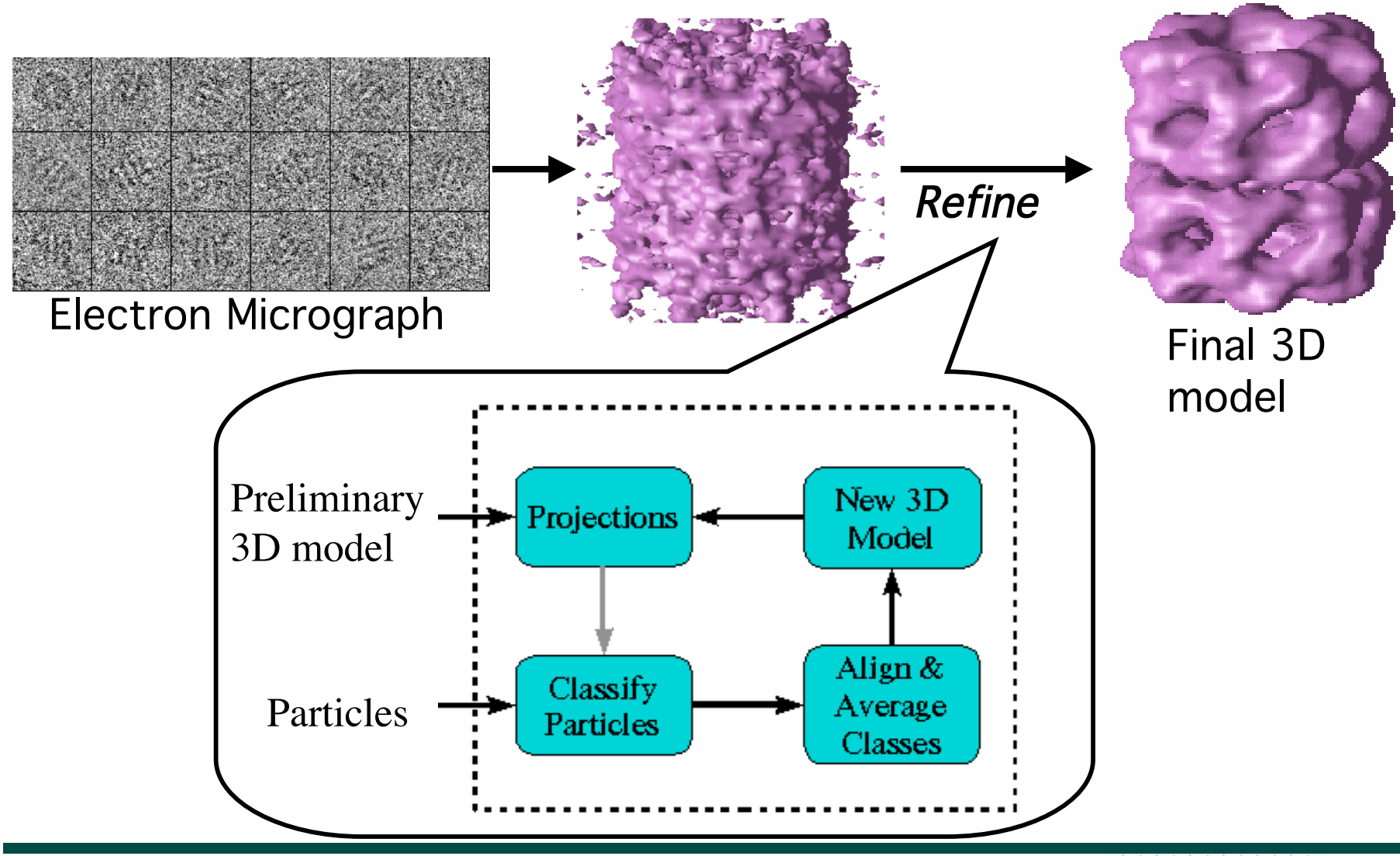
EMAN: A Workflow Application



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

VGrADS
Virtual Grid Application Development Software Project

Workflow Scheduling

- Problem: Map the components of the given Workflow DAG to a set of available Grid resources
- Objective function is to minimize the makespan of the whole Workflow application
- Overview
  - Resource Modeling using NWS and MDS
  - Sophisticated Application Component Performance models that take into account both computational performance and memory hierarchy performance
  - Walk the DAG and find the components that are currently available
  - Add data movement costs from the slowest predecessor in the performance model of the successor
  - Adapted known heuristics from domain of scheduling parameter sweep applications and use them to schedule available components

Workflow Scheduling: Results

- **Testbed**
  - 64 dual processor Itanium IA-64 nodes (900 MHz) at Rice University Terascale Cluster [RTC]
  - 60 dual processor Itanium IA-64 nodes (1300 MHz) at University of Houston [acrl]
  - 16 Opteron nodes (2009 MHz) at University of Houston Opteron cluster [medusa]
- **Experiment**
  - Ran the EMAN refinement cycle and compared running times for "classesbymra", the most compute intensive parallel step in the workflow
  - Determine the 3D structure of the 'rdv' virus particle with large input data [2GB]

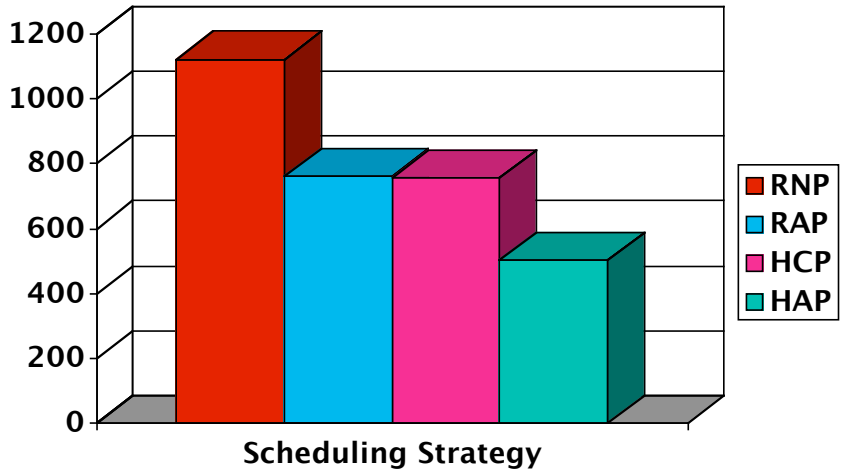
Results: Efficient Scheduling

- We compared the following workflow scheduling strategies
  1. Heuristic Scheduling with accurate performance models generated semi-automatically - HAP
  2. Heuristic Scheduling with crude performance models based on CPU power of the resources - HCP
  3. Random Scheduling with no performance models - RNP
  4. Weighted random scheduling with accurate performance models - RAP
- We compared the 'makespan' of the "classesbymra" step for the different scheduling strategies

Results: Efficient Scheduling

Scheduling method	# instances mapped to RTC (IA-64)	# instances mapped to medusa (Opteron)	# nodes picked at RTC	# nodes picked at medusa	Execution Time at RTC (minutes)	Execution Time at medusa (minutes)	Overall makespan (minutes)
HAP	50	60	50	13	386	505	505
HCP	58	52	50	13	757	410	757
RNP	89	21	43	9	1121	298	1121
RAP	57	53	34	10	762	530	762

- Set of resources: 50 RTC nodes, 13 medusa nodes
- HAP - Heuristic Accurate PerfModel
  HCP - Heuristic Crude PerfModel
  RNP - Random No PerfModel
  RAP - Random Accurate PerfModel



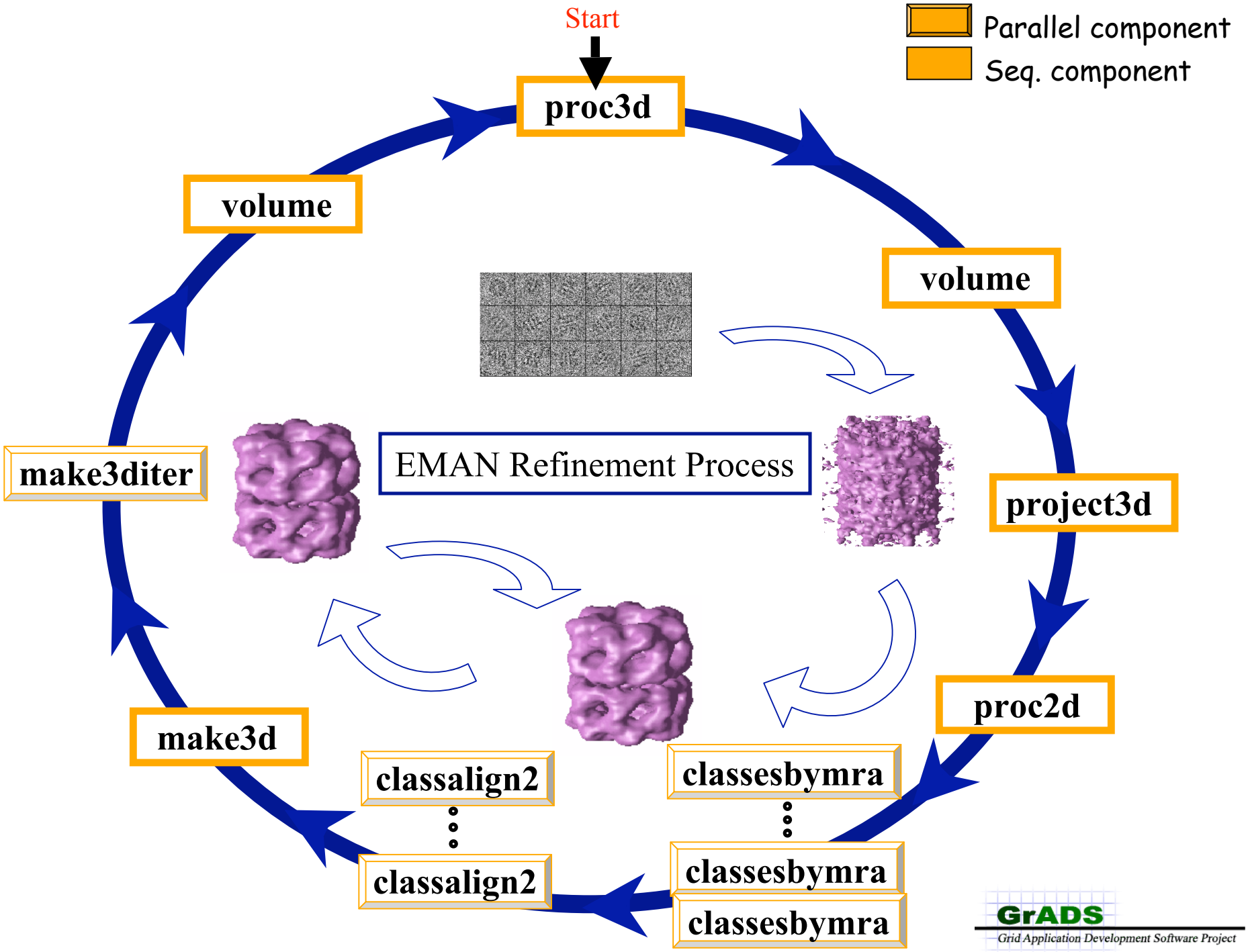
Results: Load Balance

# instances mapped to RTC (IA-64)	# instances mapped to medusa (Opteron)	# instances mapped to acrl (IA-64)	Execution Time at RTC (minutes)	Execution Time at medusa (minutes)	Execution time at acrl (minutes)	Overall makespan (minutes)
29	42	39	383	410	308	410

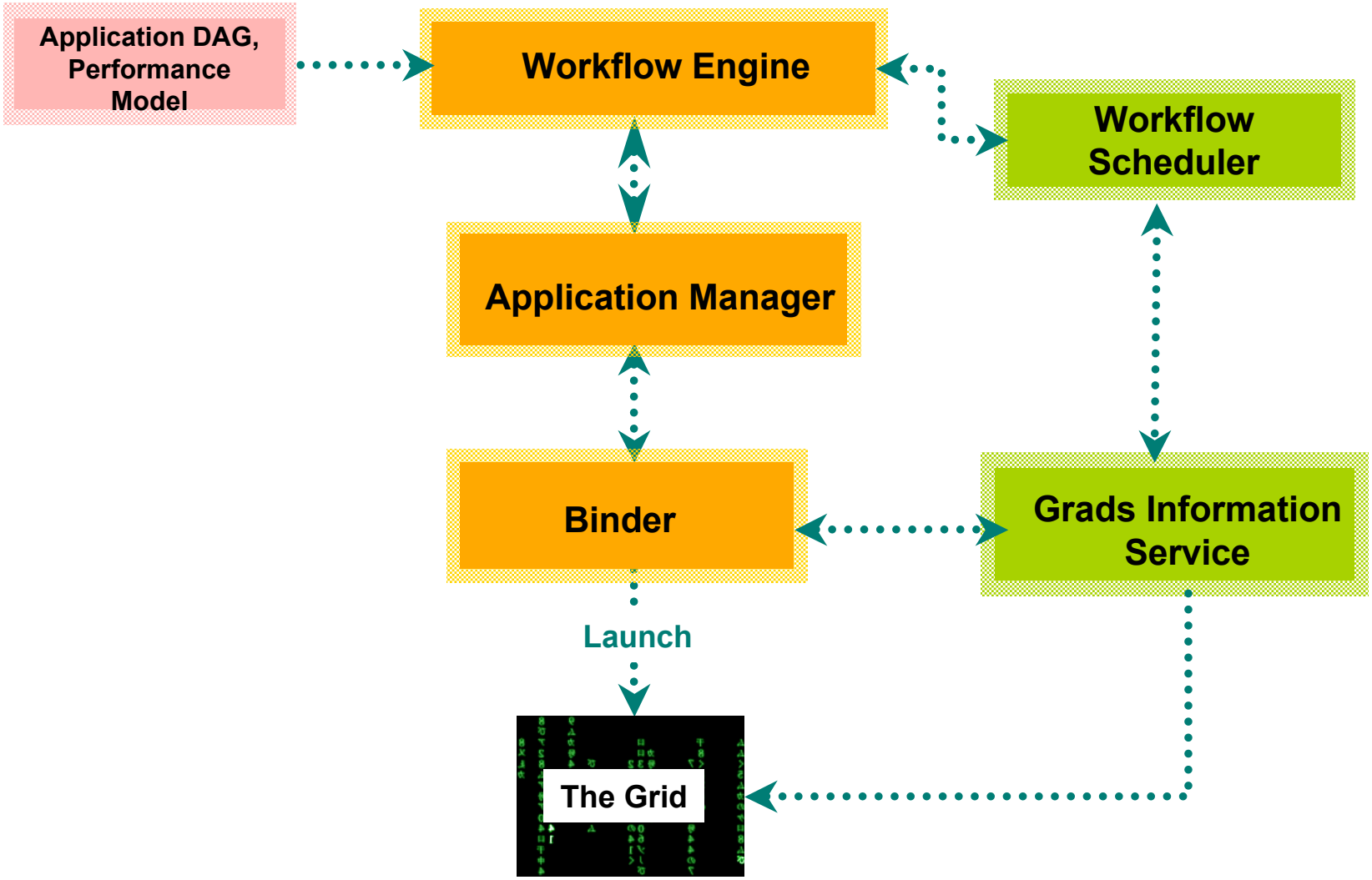
- Set of resources: 43 RTC nodes, 14 medusa nodes, 39 acrl nodes
- Good load balance due to accurate performance models

Results: Accurateness of Performance Models

- **Our performance models were pretty accurate**
  - **rank[RTC_node] / rank[medusa_node] = 3.41**
  - **actual_exec_time[RTC_node] / actual_exec_time[medusa_node] = 3.82**
  - **rank[acrl_node] / rank[medusa_node] = 2.36**
  - **actual_exec_time[acrl_node] /actual_exec_time[medusa_node] = 3.01**
- **Accurate relative performance model values result in efficient load balance of the classesbymra instances**


Start
Parallel component
Seq. component
proc3d
volume
volume
project3d
proc2d
classesbymra
classesbymra
classesbymra
classalign2
classalign2
make3d
make3diter
EMAN Refinement Process
GrADS
Grid Application Development Software Project

Framework


Application DAG, Performance Model
Workflow Engine
Workflow Scheduler
Application Manager
Binder
Grads Information Service
Launch
The Grid

Demo
follows..

Results: Accurateness of Performance Models

- Our performance models were pretty accurate
  - rank[RTC_node] / rank[medusa_node] = 3.41
    - rank[RTC_node] = 8802.31
    - rank[medusa_node] = 2578.24
  - actual_exec_time[RTC_node] / actual_exec_time[medusa_node] = 3.82
    - actual_exec_time[RTC_node] = 386 minutes
    - actual_exec_time[medusa_node] = 101 minutes
  - rank[acrl_node] / rank[medusa_node] = 2.36
    - rank[acrl_node] = 6093.91
    - rank[medusa_node] = 2578.24
  - actual_exec_time[acrl_node] /actual_exec_time[medusa_node] = 3.01
    - actual_exec_time[acrl_node] = 308 minutes
    - actual_exec_time[medusa_node] = 101 minutes
- Accurate relative performance models result in efficient load balance of the classesbymra instances