EMAN, Scheduling, Performance Prediction, and Virtual Grids

Charles Koelbel
chk@cs.rice.edu

http://vgrads.rice.edu/site_visit/april_2005/slides/koelbel

VGrADS
Virtual Grid Application Development Software Project

Credits

- Baylor College of Medicine - EMAN research group
  - Wah Chiu, Director - National Center for Macromolecular Imaging
  - Steve Ludtke, Principal author
  - Wen Jiang, Liwei Peng, Phil Baldwin, Shunming Fang, Htet Khant, Laurie Nason
- Rice University - VGrADS group
  - Ken Kennedy and Chuck Koelbel, Principal Investigators
  - Mark Mazina, Research Staff
  - Anirban Mandal, Anshu DasGupta, Gabriel Marin, Ryan Zhang
- University of Houston - VGrADS group
  - Lennart Johnsson, Principal Investigator
  - Bo Liu, Mitul Patel
- University of Southern California - VGrADS Group
  - Carl Kesselman, Principal Investigator
  - Gurmeet Singh
- University of California, San Diego - VGrADS Group
  - Andrew A. Chien and Henri Casanova, Principal Investigators
  - Yang-suk Kee, Postdoc
  - Jerry Chou, Richard Huang, Dennis Logothetis

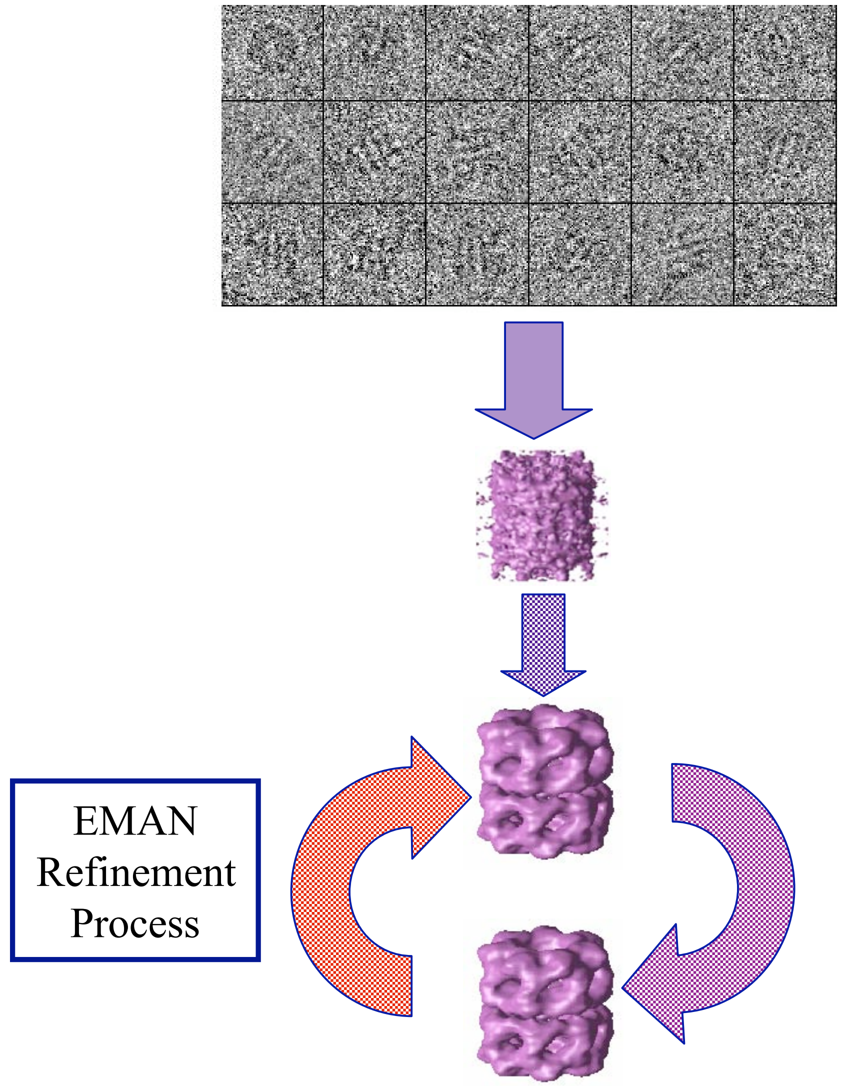
Outline

- **Overview of EMAN ❶**
- **Scheduling EMAN execution ❶**
- **Predicting EMAN performance ❶ ❷**
- **Future directions ❸ ❹**

- **Related posters**
  - ❶ "Performance Model-Based Scheduling of EMAN Workflows" by Anirban Mandal (Rice) and Bo Liu (U Houston)
  - ❷ "Scalable Cross-Architecture Predictions of Memory Latency for Scientific Applications" by Gabriel Marin (Rice)
  - ❸ "Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments" by Richard Huang (UCSD)
  - ❹ "Optimizing Grid-Based Workflow Execution" by Gurmeet Singh (ISI)

EMAN - Electron Micrograph Analysis

- **Software for Single Particle Analysis and Electron Micrograph Analysis**
  - Open source software for the scientific community
  - Developed by Wah Chiu & Steve Ludtke, Baylor College of Medicine
  - http://ncmi.bcm.tmc.edu/homes/stevel/EMAN/EMAN/doc/
- **Performs 3-D reconstruction of a particle from randomly-oriented images**
  - Typical particle = Virus or ion channel
  - Typical images = Electromicrographs
  - Typical data set = 10K-100K particles
  - Useful for particles about 10-1000nm
- **GrADS/VGrADS project to put EMAN on Grid**

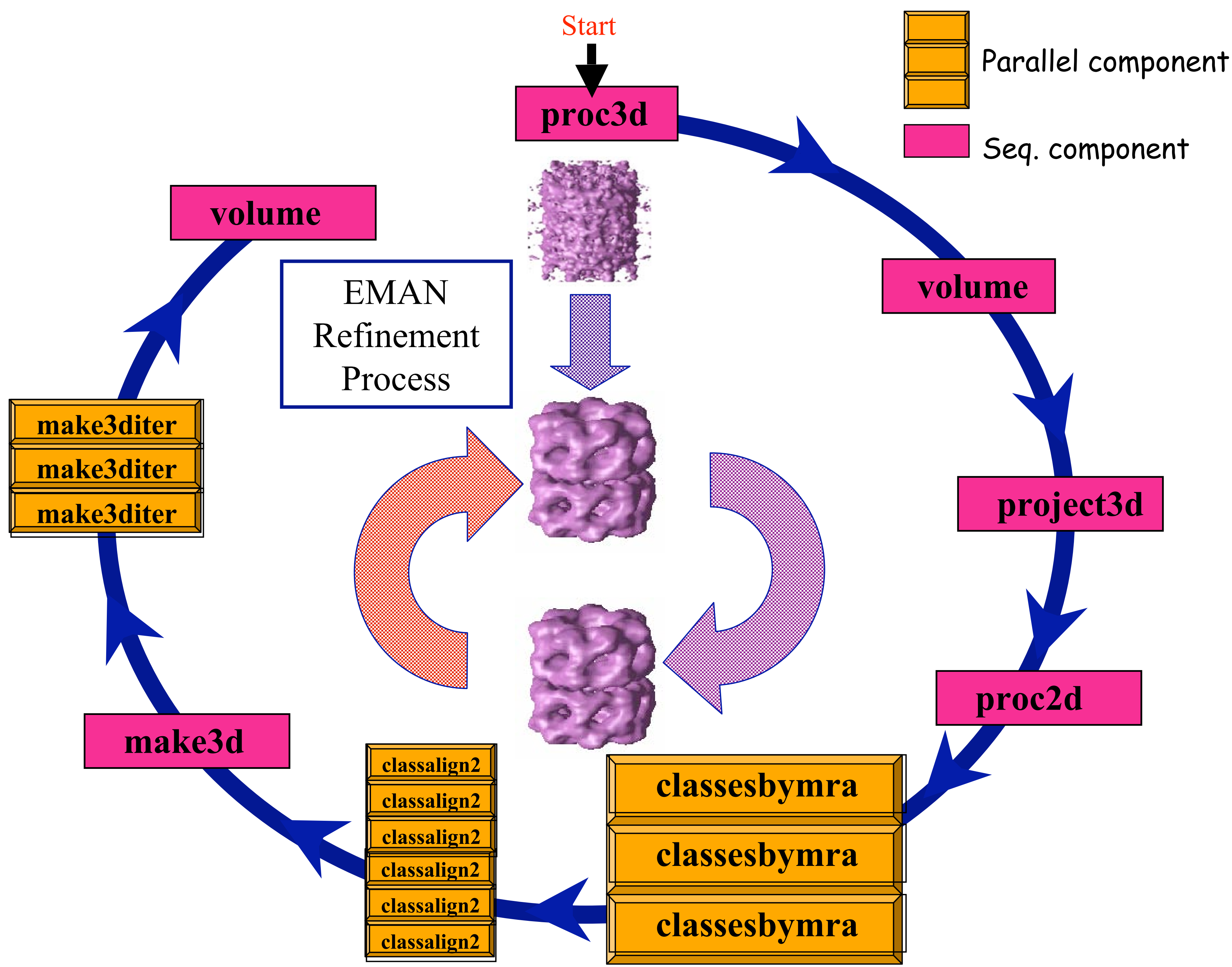


All electron micrograph and 3-D reconstruction images courtesy of *Wah Chiu* & *Steven Ludtke,* Baylor College of Medicine

VGrADS
Virtual Grid Application Development Software Project

EMAN from a CS Viewpoint

- EMAN is a great workflow application for VGrADS
  - Represented as a task graph
  - Heterogeneous tasks, some parallel & some sequential
  - Parallel phases are parameter sweeps well-suited to parallelism
  - Implemented with Python calling C/C++ modules (now)
- Technical issues
  - Computational algorithms for guiding the refinement
    - Currently fairly brute-force, subtler algorithms under development
  - Scheduling task graph on heterogeneous resources
    - Computation cost depends on processor characteristics, availability
    - Communication cost depends on network characteristics, file systems
    - We want pre-computed schedules (on-line schedules = future work)
  - And many, many, many little details
- More detail in poster session
  - ➊ "Performance Model-Based Scheduling of EMAN Workflows" by Anirban Mandal (Rice) and Bo Liu (UH)


Start
proc3d
Parallel component
Seq. component
volume
volume
EMAN
Refinement
Process
project3d
make3diter
make3diter
make3diter
proc2d
make3d
classalign2
classalign2
classalign2
classalign2
classalign2
classalign2
classesbymra
classesbymra
classesbymra

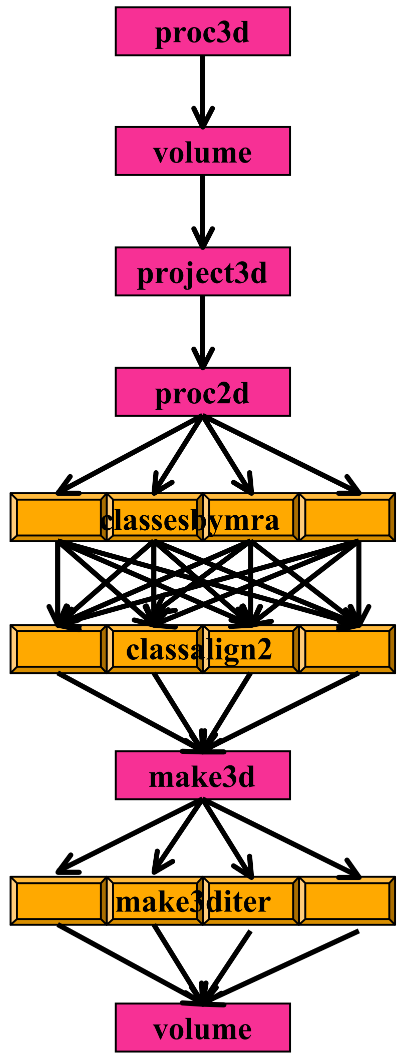
Outline

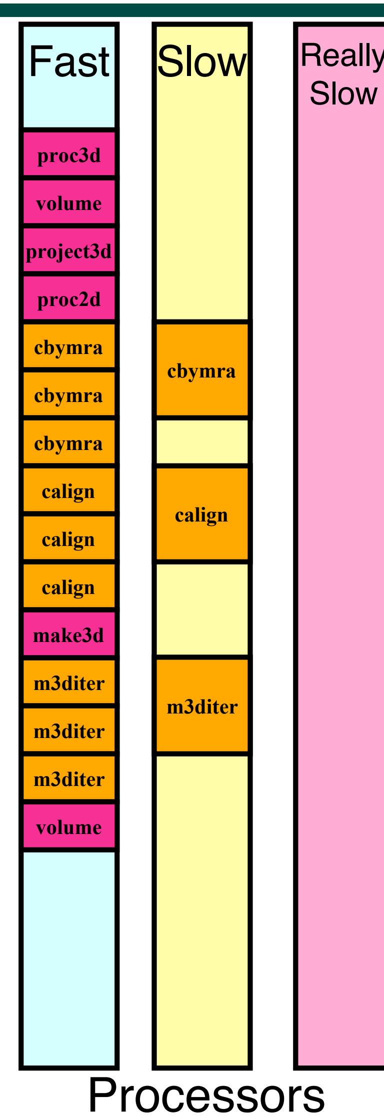
- Overview of EMAN
- Scheduling EMAN execution
- Predicting EMAN performance
- Future directions

Heuristic Scheduling Algorithm

foreach *heuristic* **do**
  while *all components not mapped* **do**
    Find available comps;
    Calculate rank(comp,R) for all comps,R;
    findBestSchedule(comps,heuristic);
  endwhile
endforeach
Select heuristic with minimum makespan;
Output selected mapping;

findBestSchedule(*comps,h*))
  while *all comps not mapped* **do**
    foreach *Component, C* **do**
      foreach *Resource, R* **do**
        ECT(C,R)=rank(C,R)+EAT(R);
      endforeach
      Find minECT(C,R) over all R;
      Find 2nd_minECT(C,R) over all R;
    endforeach
    if (h==min-min) $j^* = j_1$ with min(minECT(j_1,R));
    if (h==max-min) $j^* = j_2$ with max(minECT(j_2,R));
    if (h==sufferage) $j^* = j_3$ with
      min(2nd_minECT(j_3,R)-minECT(j_3,R));
    Store mapping for j*;
    Update EAT(R) and makespan;
  endwhile





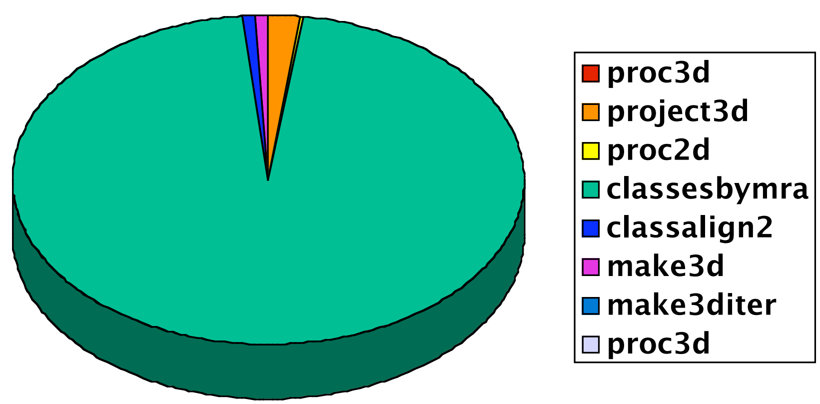
❶ **"Performance Model-Based Scheduling of EMAN Workflows" by Anirban Mandal (Rice) and Bo Liu (UH)**

VGrADS
Virtual Grid Application Development Software Project

EMAN Scheduling: Large Data, Small Grid

EMAN component	# instances mapped to i2 (IA-64)	# instances mapped to torc (IA-32)	# nodes picked at i2	# nodes picked at torc	Execution time at i2 (min)	Execution time at torc (min)	Overall makespan (min)
proc3d	1		1		1		1
project3d	1		1		108		108
proc2d	1		1		1		1
classesbymra	68	42	6	7	5070	4901	5070
classalign2	379	0	6	0	45	0	45
make3d	1		1		47		47
make3diter	1	0	1	0	1	0	1
proc3d	1		1		1		1

- Set of resources:
  - 6 i2 nodes at U of Houston (IA-64)
  - 7 torc nodes at U of Tennessee @ Knoxville (IA-32)
- Data set: RDV
  - Medium/large (2GB)
- Key was load-balancing classesbymra component using performance models

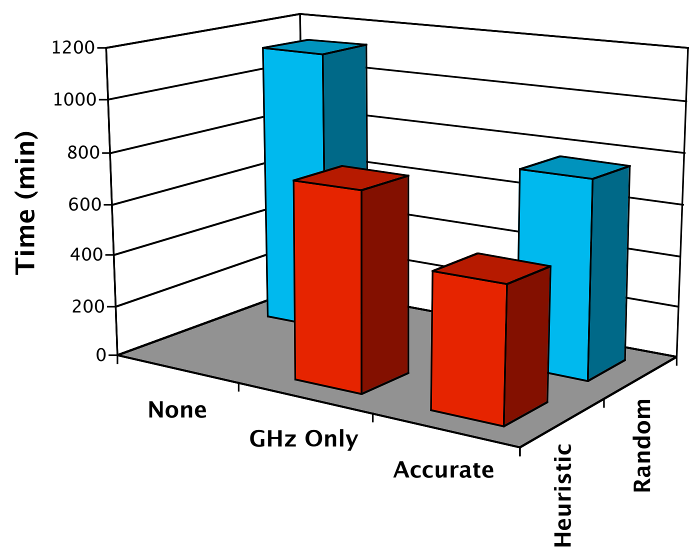


Hereafter, we only show classesbymra in the timings

EMAN Scheduling: Varying Performance Models

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 (min)	Execution time at medusa (min)	Overall makespan (min)
RNP	89	21	43	9	1121	298	1121
RAP	57	53	34	10	762	530	762
HGP	58	52	50	13	757	410	757
HAP	50	60	50	13	386	505	505

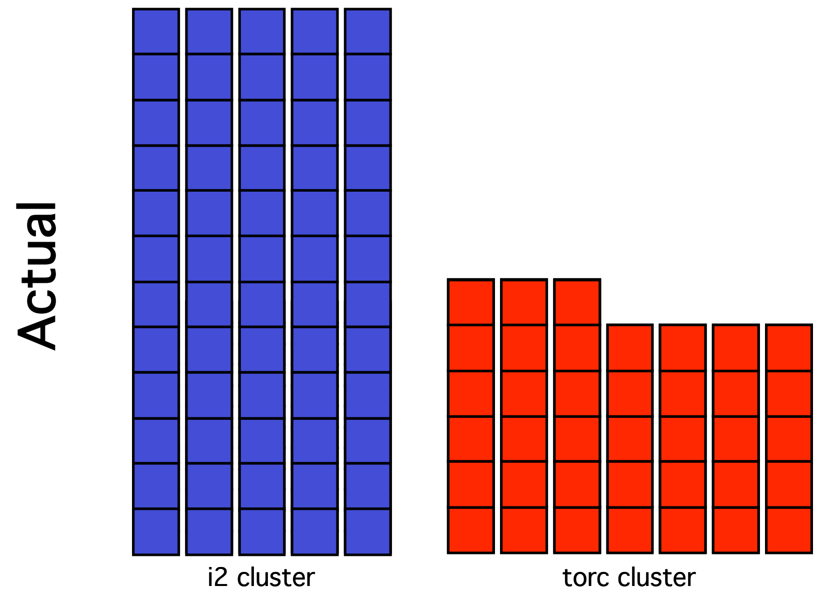
- Set of resources:
  - 50 rtc nodes at Rice (IA-64)
  - 13 medusa nodes at U of Houston (Opteron)
- RDV data set
- Varying scheduling strategy
  - RNP - Random / No PerfModel
  - RAP - Random / Accurate PerfModel
  - HGP - Heuristic / GHz Only PerfModel
  - HAP - Heuristic / Accurate PerfModel



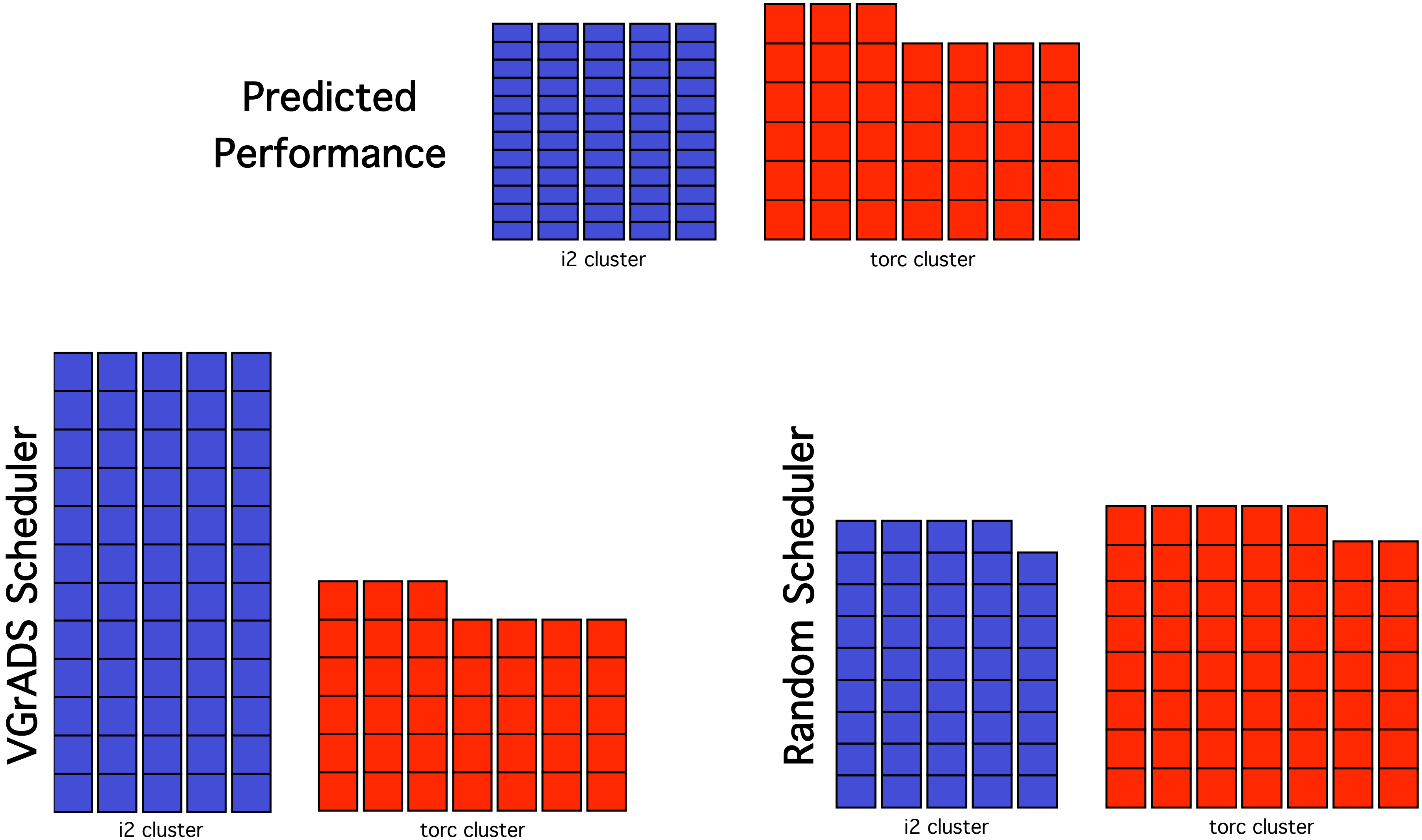
EMAN Scheduling: Small Data, Small Grid

Run	# instances mapped to i2 (IA-64)	# instances mapped to torc (IA-32)	Execution time on i2 (min:sec)	Execution time on torc (min:sec)	Overall makespan (min:sec)
VGrADS heuristics	60	38	16:41	7:51	16:41
Random placement	44	54	9:38	10:28	10:28

- Set of resources:
  - 5 i2 nodes at U of Houston (IA-64)
  - 7 torc nodes at U of Tennessee (IA-32)
  - All nodes used
- GroEl data set
  - 200MB
- Major load imbalance
  - External load on i2 nodes invalidated VGrADS performance model
  - Random scheduler too dumb to notice



EMAN Scheduling: Predicted and Actual Load Balance



Outline

- Overview of EMAN
- Scheduling EMAN execution
- **Predicting EMAN performance**
- **Future directions**

EMAN Performance Modeling

- **Rank of a component is total time to run it on a resource**

$$Rank(comp_i, res_j) = EstExecTime_i(size(comp_i), arch(res_j)) + EstCommTime(comp_i, res_j)$$

- **Execution time is computation time and memory access times**

$$EstExecTime(n,a) = \frac{FP(n,a) + L_1(n,a) + L_2(n,a) + L_3(n,a)}{Clock(a)}$$

$$FP(n,a) = FPcount(n) \times \frac{1 + FPstalled(n,a)}{FPpipes(a)}$$

$$L_k(n,a) = L_k count(n) \times L_k penalty(a), \quad k = 1,2,3$$

- **Communication time is latency plus bandwidth cost**
  - **Estimated from NWS**

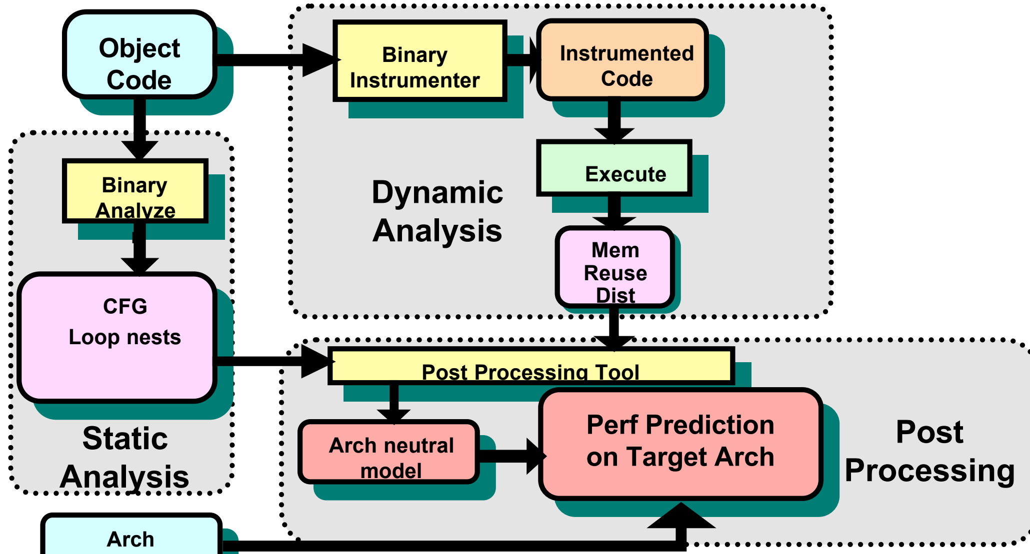
$$EstCommTime(c,r) = \sum_{p \in Parent(c)} \left(Lat(map(p), r) + Vol(p,c) \cdot BW(map(p), r) \right)$$

EMAN Performance Modeling: Computation Time (FP)

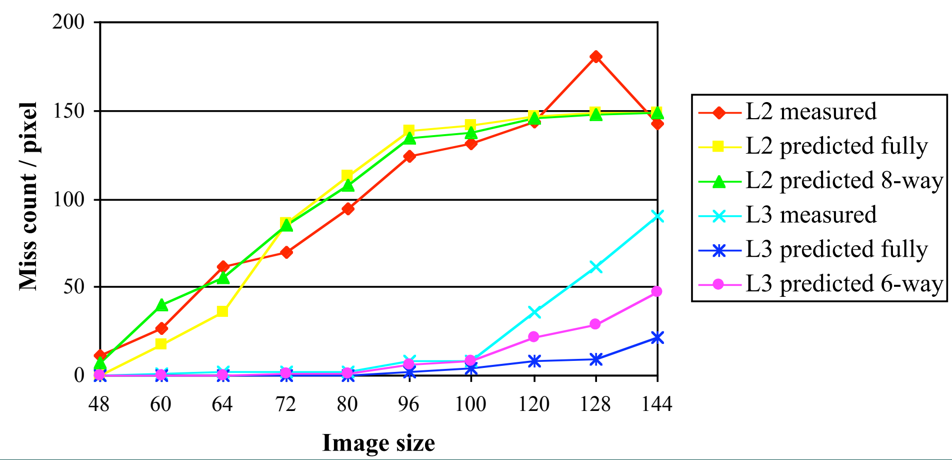
- **(Floating point) Computation time is estimated from semi-empirical models**
  - Form of model given by application experts
    - EMAN is floating-point intensive □ Count floating-point ops
    - Classesbymra is based on FFT □ is $O(n^2+n^2 \log_2(n))$ □
      Fit to $c_5 \cdot n^2 \cdot \log_2(n) + c_4 \cdot n^2 + c_3 \cdot \log_2(n) + c_2 \cdot n + c_1$
  - Training runs with small data sizes
  - Collect floating-point operation counts from hardware performance counters
  - Least-squares fit of collected data to model to determine coefficients (FPcount, FPstalled)
  - Architecture parameters used to complete model (FPpipes)

EMAN Performance Modeling: Memory Access Time (L_1, L_2, L_3)

- **Memory access time (cache miss penalty) is estimated from black-box analysis of object code**
  - Static analysis determines code structure
  - Training runs with instrumented binary produce architecture-independent memory reuse distance histograms
  - Fit polynomial models of reuse distances and number of accesses
  - Convolve with architecture features (e.g. cache size) for full model



project3d



❷ "Scalable Cross-Architecture Predictions of Memory Latency for Scientific Applications" by Gabriel Marin (Rice)

Accuracy of EMAN Performance Models

- **Machine-neutral performance prediction models were accurate on unloaded systems**
  - Combining application knowledge, static analysis, dynamic instrumentation gave accurate results
    - Good case: rank[RTC] / rank[medusa] = 3.41; actual_time[RTC] / actual_time[medusa] = 3.82
    - Less good case: rank[acrl] / rank[medusa] = 2.36; actual_time[acrl] /actual_time[medusa] = 3.01
  - Caveat: It's still an art, not a science
- **Adjustment is required for (possibly) loaded systems**
  - NWS load predictions should provide an appropriate scaling factor

Outline

- Overview of EMAN
- Scheduling EMAN execution
- Predicting EMAN performance
- **Future directions**

EMAN Lessons for Virtual Grids

- **Scheduling support is important**
  - Requires performance information from vgES
  - Would benefit from performance guarantees from vgES
- **Resource selection is key**
  - New VG request allows good resource provisioning …
  - … if you know what you want
    - Great topic for a thesis
- **Scalability requires new thinking**
  - Hierarchy of VGs should be helpful
  - Virtual grid summarization allows scalable information collection
    - But we still need algorithms to take advantage of vg

Ongoing Research

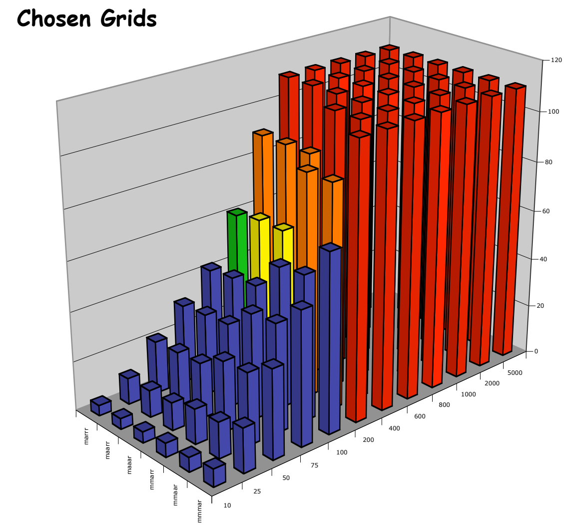
- **Multi-level scheduling**
  - Rice / UCSD collaboration
  - Separate concerns between resource selection and mapping
  - Key question: Do we lose information, and if so how much?
  - ❸ "Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments" by Richard Huang
- **Application management**
  - ISI / Rice / UCSD collaboration
  - Leverage Pegasus framework for workflow management, optimization, …
  - Key question: How do we separate concerns?
  - ❹ "Optimizing Grid-Based Workflow Execution" by Gurmeet Singh (ISI)
- **Scripting language support**
  - Rice project
  - Telescoping languages tie-in
  - Key question: How can we leverage high-level language/application knowledge in a Grid environment?

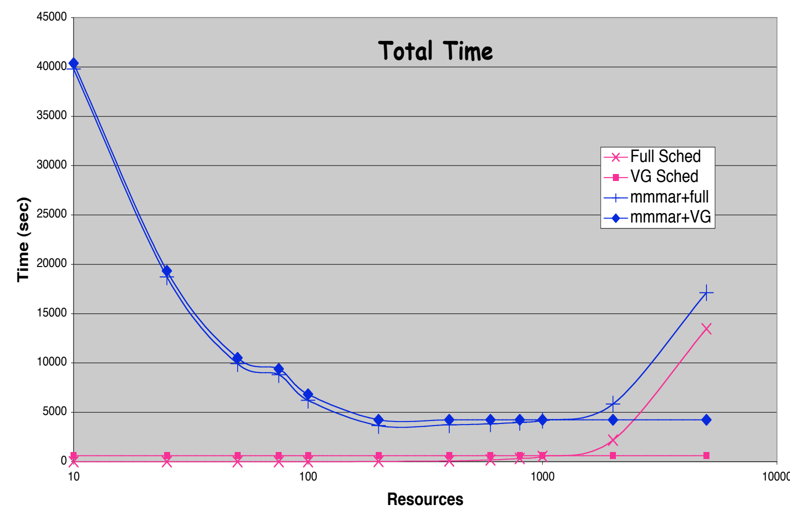
Multi-level Scheduling

- **Current VGrADS scheduler is limited**
  - O(components*resources) complexity limits scalability
  - Look-ahead scheduling limited
- **vgES offers improvements**
  - Separate concerns between resource selection and resource mapping
  - Fast VG Finding reduces universe of resources to search
  - VG Binding limits uncertainty and complexity
    - Provide performance guarantees
  - Natural hierarchy of schedulers
    - Schedule work between clusters
    - Schedule work within a cluster (perhaps recursively)
- **But…**
  - Can we select the best resources without scheduling them?

Multi-level Scheduling: An Illustration

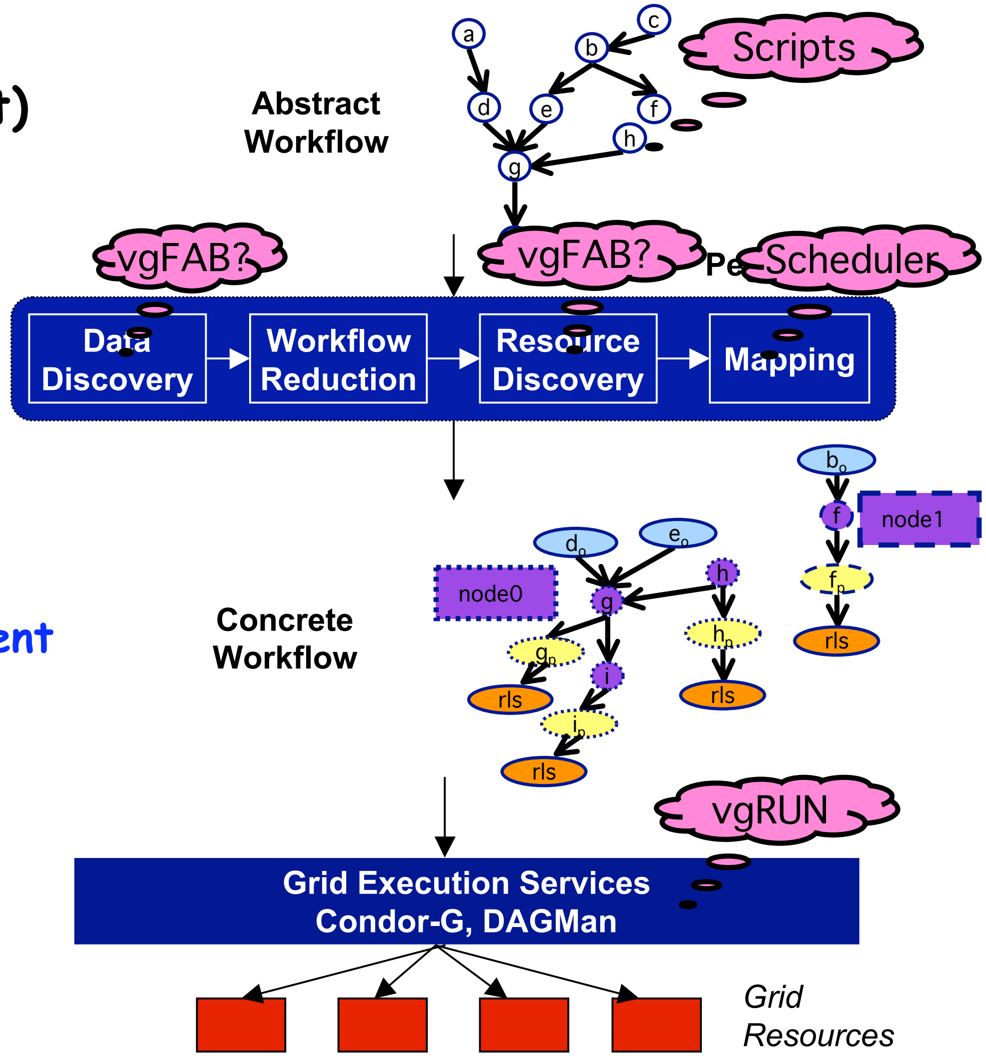
- **First experiment**
  - Schedule EMAN with rdv data on notional (large) grid using VGrADS scheduler
  - Generate VG and schedule on it
  - Compute total time for each
    - Wall-clock time for scheduler
    - Predicted makespan for computation
  - Repeat for many grids
- **Partial, preliminary results**
  - Ran out of time to integrate vgES and scheduler
  - Still clearly shows limits of full scheduler on large grid





Application Management

- We are experimenting with Pegasus (from GriPhyN project) as a framework for EMAN
  - http://pegasus.isi.edu/
- Pegasus supports
  - Workflow execution based on "abstract" DAGs
  - Data discovery, replica management, computation scheduling, and data management
  - Fault tolerance and component launch (through DAGMAN and Condor-G)
- Pegasus needs
  - Link to vgES



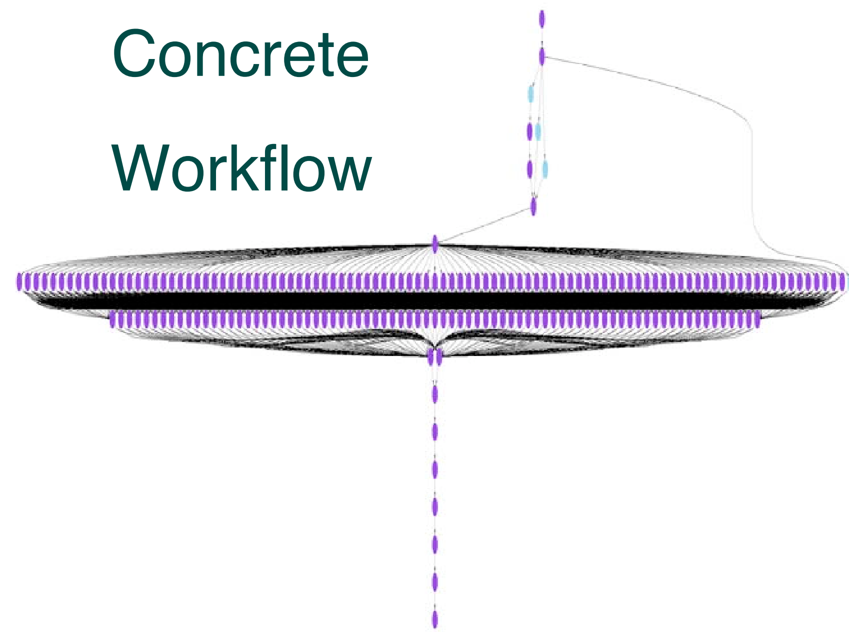
Application Management: First Experiments

- **Successfully ran EMAN with GroEl data under Pegasus**
  - **Create abstract workflow (XML file) manually from EMAN script**
  - **Generated concrete workflow (Condor submit files) using Pegasus**
  - **Executed on ISI Condor pool (20 machines)**
  - **Now porting to Teragrid**
    - **Same abstract workflow, but new binaries needed**


```
- <job id="ID000001" name="proc3d" level="1" dv-name="proc3d_1">
-       <argument><filename file="threed.0a.mrc" /> <filename file="x.0.mrc" />
        clip=84,84,84 mask=42  </argument>
        <uses file="threed.0a.mrc" link="input" dontRegister="false"
        dontTransfer="false" />
        <uses file="x.0.mrc" link="output" dontRegister="true" dontTransfer="true" />
</job>
- <job id="ID000002" name="volume" level="2" dv-name="volume_2">
-       <argument><filename file="x.0.mrc" />  2.800000 set=800.000000  </argument>
        <uses file="x.0.mrc" link="inout" dontRegister="true" dontTransfer="true" />
</job>
- <job id=.......
```


Concrete

Workflow



Abstract Workflow