Virtual Grid Application Development Software (VGrADS),
Electron Microscopy Analysis (EMAN),
and Other Acronyms

Charles Koelbel
Rice University
chk@cs.rice.edu

http://vgrads.rice.edu/
Virtual Grid Application Software Development (VGrADS)

• VGrADS is an NSF-funded Information Technology Research project

Rich Wolski
Fran Berman
Andrew Chien
Henri Casanova

Keith Cooper
Ken Kennedy
Charles Koelbel
Richard Tapia
Linda Torczon

Carl Kesselman
Dan Reed

Jack Dongarra
Lennart Johnsson

Plus many graduate students, postdocs, and technical staff!
The VGrADS Vision: National Distributed Problem Solving
The VGrADS Vision: Distributed Problem Solving
VGrADS Vision

• Build a National Problem-Solving System on the Grid
  – Transparent to the user, who sees a problem-solving system

• Why don’t we have this today?
  – Poor programming tools and models
    - Tied to physical resources
    - If programming is hard, the Grid will not reach its potential
  – Complex programming problems
    - Dynamic resources require adaptivity
    - Unreliable resources require fault tolerance
    - Uncoordinated resources require management

• What do we need?
  – (Based on previous GrADS project - http://www.hipersoft.rice.edu/grads/)
  – A more abstract view of the Grid
    - Each developer sees a specialized “virtual grid”
  – Simplified programming models built on the abstract view
    - Permit the application developer to focus on the problem
A Virtual Grid (VG) takes
- Shared heterogeneous resources
- Scalable information service

and provides
- An hierarchy of application-defined aggregations (e.g. ClusterOf) with constraints (e.g. processor type) and rankings

Virtual Grid Execution System (vgES) implements VG
- VG Definition Language (vgDL)
- VG Find And Bind (vgFAB)
- VG Monitor (vgMON)
- VG Application Launch (VgLAUNCH+DVCW)
- VG Resource Info (vgAgent)
Tools: Scheduling and Fault Tolerance Methods

VGrADS is studying a range of tools for grid programming tasks, including:

- **Scheduling of workflow computations**
  - Off-line look-ahead scheduling dramatically improves in makespan (total time)
  - Accurate performance models significantly affect quality of scheduling
  - Queue wait prediction allows scheduling into batch queues

- **Fault tolerance**
  - Diskless checkpointing for linear algebra computations (application-specific)
  - Temporal reasoning for fault prediction
  - Optimal checkpoint frequency for iterative applications

![Image of online vs. offline scheduling on a heterogeneous platform](image1)

![Image of performance models and schedulers on heterogeneous platforms](image2)

![Diagram of diskless checkpointing with processors and parity processor](image3)
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/st Evel/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
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
EMAN Refinement Process

Start

proc3d

Volume

Parallel component

Seq. component

make3d

make3diter

make3diter

make3diter

project3d

classalign2

classalign2

classalign2

classalign2

classalign2

classesbymra

classesbymra

classesbymra

classesbymra

classesbymra

proc2d
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 minECT(C,R) over all R;
      endforeach
      if (h==min-min) j* = j1 with min(minECT(j1,R));
      if (h==max-min) j* = j2 with max(minECT(j2,R));
      if (h==sufferage) j* = j3 with
      Endforeach
      if (h==min-min) j* = j1 with min(minECT(j1,R));
      if (h==max-min) j* = j2 with max(minECT(j2,R));
      if (h==sufferage) j* = j3 with
      Store mapping for j*;
      Update EAT(R) and makespan;
    endwhile

Processes
EMAN Scheduling: Varying Performance Models

<table>
<thead>
<tr>
<th>Scheduling method</th>
<th># instances mapped to rtc (IA-64)</th>
<th># instances mapped to medusa (Opteron)</th>
<th># nodes picked at rtc</th>
<th># nodes picked at medusa</th>
<th>Execution time at rtc (min)</th>
<th>Execution time at medusa (min)</th>
<th>Overall makespan (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNP</td>
<td>89</td>
<td>21</td>
<td>43</td>
<td>9</td>
<td>1121</td>
<td>298</td>
<td>1121</td>
</tr>
<tr>
<td>RAP</td>
<td>57</td>
<td>53</td>
<td>34</td>
<td>10</td>
<td>762</td>
<td>530</td>
<td>762</td>
</tr>
<tr>
<td>HGP</td>
<td>58</td>
<td>52</td>
<td>50</td>
<td>13</td>
<td>757</td>
<td>410</td>
<td>757</td>
</tr>
<tr>
<td>HAP</td>
<td>50</td>
<td>60</td>
<td>50</td>
<td>13</td>
<td>386</td>
<td>505</td>
<td>505</td>
</tr>
</tbody>
</table>

- **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 Performance Modeling

- **Rank of a component is total time to run it on a resource**
  \[ \text{Rank}(\text{comp}_i, \text{res}_j) = \text{EstExecTime}_i(\text{size}(\text{comp}_i), \text{arch}(\text{res}_j)) + \text{EstCommTime}(\text{comp}_i, \text{res}_j) \]

- **Execution time is computation time and memory access times**
  \[ \text{EstExecTime}(n,a) = \frac{\text{FP}(n,a) + L_1(n,a) + L_2(n,a) + L_3(n,a)}{\text{Clock}(a)} \]
  \[ \text{FP}(n,a) = \text{FPcount}(n) \times \frac{1 + \text{FPstalled}(n,a)}{\text{FPpipes}(a)} \]
  \[ L_k(n,a) = L_k\text{count}(n) \times L_k\text{penalty}(a), \quad k = 1, 2, 3 \]

- **Communication time is latency plus bandwidth cost**
  - Estimated from NWS
  \[ \text{EstCommTime}(c,r) = \sum_{p \in \text{Parent}(c)} (\text{Lat}(\text{map}(p),r) + \text{Vol}(p,c) \cdot \text{BW}(\text{map}(p),r)) \]
EMAN Performance Modeling

- **FP(n,a)** is estimated from semi-empirical models
  - Form of model given by application experts
    - Classes by mra based on FFT
    - Fit to \( c_5 \cdot n^2 \cdot \log_2(n) + c_4 \cdot n^2 + c_3 \cdot \log_2(n) + c_2 \cdot n + c_1 \)

- **L_k(n,a)** 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
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
Random Thoughts After Wine & Cheese

• I've talked only about technical issues
  — Social / political / administration problems still dominate
  — Keeping EMAN interface crucial to keeping collaboration
    - But it wasn’t enough for practical use
  — Rewarding both sides of collaboration needed
    - Fortunately, Ken Kennedy and Wah Chiu are both visionaries
    - Unfortunately, funding agencies aren’t (yet)

• Information Technology implementation took resources
  — People time (at grad student wages)
  — Computer resources (from other projects)
  — Data and network (sidestepped real issues)

• Computer Science / Computational Science research still needed
  — Must be incremental (or incrementally implemented) for production
    - E.g. Get EMAN working under Condor-G, then add scheduling