GPU-enabled Studies of Molecular Systems on Keeneland

On pursuing high resource utilization and coordinated simulations’ progression

Michela Taufer, Sandeep Patel, Samuel Schlachter, and Stephen Herbein
University of Delaware

Jeremy Logan, ORNL
Taxonomy of simulations

• Simulations applying fully atomistically resolved molecular models and force fields
  ▪ GPUs enable longer time and space scales
• Variable job lengths (ns/day):
  ▪ As a trajectory evolves
  ▪ Across trajectories with different e.g., concentrations
• Fully or partially coordinated simulation progression:
  ▪ Fully coordinated needed for e.g., replica-exchange molecular dynamics (REMD)
  ▪ Partially coordinated for e.g., SDS and nanotubes systems
Constraints on high–end computer systems

• Resource constraints on high-end clusters:
  ▪ Limited wall-time limit per job (e.g., 24 hours)
  ▪ Mandatory use of resource managers
  ▪ No direct submission and monitoring of GPU jobs

• Logical GPU job does not map to physical GPU job
  ▪ Workflow managers still in infancy

• System and application failures on GPUs are undetected
  ▪ Resource managers remain with no notion of job terminations on GPUs
Moving beyond virtualization

• When clusters **do include virtualization**
  ▪ E.g., Shadowfax
• We can schedule isolated CPU/GPU pairs
  ▪ This allows us to associate failures with a specific GPU
• Virtualization imposes overheads
  ▪ Power
  ▪ Performance
  ▪ Noise or jitter
  ▪ Portability and maintainability

... and may not be available

**Our goal**: Pursuing BOTH high accelerators’ utilization and (fully or partially) coordinated simulations’ progression on GPUs in effective and cross-platform ways
Our approach

• Two software modules that plug into existing resource managers and workflow managers
  ▪ No virtualization to embrace diverse clusters and programming languages

• A companion module:
  ▪ Runs on the head node of the cluster
  ▪ Accepts jobs from workflow manager
  ▪ Instantiates "children" wrapper modules
  ▪ Dynamically splits jobs and distributes job segments to wrapper modules

• A wrapper module:
  ▪ Launches on compute node as a resource manager job
  ▪ Receives and runs job segments from companion module
  ▪ Reports status of job segments to companion module
Modules in action

User node

Workflow Manager

Companion Module

Resource Manager

Front-end node

Job queue

Back-end node
Modules in action

User node

Workflow Manager

• generate set of 24-hour jobs

Companion Module

Resource Manager

Job queue

Front-end node

Back-end node
Modules in action

User node

Workflow Manager
- 24-hour jobs

Companion Module
- WM instance

Resource Manager
- Job queue

Front-end node

Workflow Manager:
- send set of 24-hour jobs to companion module

Companion Module:
- receive 24-hour jobs
- generate a Wrapper Module (WM) instance per back-node

Back-end node
Modules in action

User node

Workflow Manager

Companion Module

• submit WM instance as a job to resource manager

Companion Module:

24-hour jobs
WM instance

Resource Manager

Job queue

Front-end node

Back-end node
Modules in action

User node

Workflow Manager

Companion Module
- submit WM instance as a job to resource manager

Resource Manager

Job queue

Front-end node

Back-end node
Modules in action

User node

Workflow Manager

Companion Module

Resource Manager:
- launch WM instance as a job on back-end node

Front-end node

Resource Manager
- Job queue

Back-end node

WM job

24-hour jobs

WM instance
Modules in action

Wrapper Module:
• ask companion module for job segments, as many as the available GPUs

User node

Wrapper Module:

Companion Module

Resource Manager

Front-end node

Back-end node

24-hour jobs

Job queue

WM job
Modules in action

User node

Workflow Manager

Companion Module:
• fragment jobs into 6-hour subjobs

Companion Module

24-hour jobs

Resource Manager

Job queue

Front-end node

Back-end node

WM job
Modules in action

User node

Workflow Manager

Companion Module

• fragment jobs into 6-hour subjobs
• send bundle of 3 subjobs to WM job

Resource Manager

Job queue

Front-end node

Companion Module:

24-hour jobs

Back-end node

WM job
Modules in action

User node

Workflow Manager

Companion Module:  
- fragment jobs into 6-hour subjobs  
- send bundle of 3 subjobs to WM job

Companion Module:

24-hour jobs

Resource Manager

Job queue

Front-end node

WM job

Back-end node
Modules in action

User node

Workflow Manager

Companion Module

Resource Manager

Front-end node

Wrapper Module:
- instantiate subjobs on GPUs
- monitor system and application failures as well as time constraints

Back-end node
Modules in action

User node

Workflow Manager

Wrapper Module:
• instantiate subjobs on GPUs
• monitor system and application failures as well as time constraints

Companion Module

24-hour jobs

Resource Manager

Job queue

Front-end node

Back-end node
Modules in action

Wrapper Module:
- if subjob terminates prematurely because of e.g., system or application failures, it request new subjob

User node

Workflow Manager

Companion Module
- 24-hour jobs

Resource Manager
- Job queue

Front-end node

Back-end node

WM job
Modules in action

User node

Workflow Manager

Companion Module

Resource Manager

Front-end node

Back-end node

Companion Module:
• adjust length of new subjob based on heuristics, e.g., to complete initially 6-hour period
• send subjob to wrapper module for execution
Modules in action

User node

Workflow Manager

Companion Module

• adjust length of new subjob based on heuristics, e.g., to complete initially 6-hour period
• send subjob to wrapper module for execution

Resource Manager

24-hour jobs

Job queue

Front-end node

Back-end node

Companion Module:

WM job
MD Simulations

• MD simulations:
  ▪ Case study 1: Study of sodium dodecyl sulfate (SDS) molecules aqueous solutions and electrolyte solutions
  ▪ Case study 2: Study of nanotubes in aqueous solutions and electrolyte solutions

• GPU code FEN ZI (Yun Dong de FEN ZI = Moving MOLECULES)
  ▪ MD simulations in NVT and NVE ensembles and energy minimization in explicit solvent
  ▪ Constraints on interatomic distances e.g., shake and rattle, atomic restraints, and freezing fast degrees of motions
  ▪ Electrostatic interactions, i.e., Ewald summation, performed on GPU

• Metrics of interest:
  ▪ Utilization of GPUs – i.e., time ratio accountable for simulation’s progression
The Keeneland system

• GPU description:
  ▪ 3 M2090 GPUs per node

• Software:
  ▪ TORQUE Resource Manager
  ▪ Globus allows for the use of Pegasus Workflow Manager
  ▪ Shared Lustre file system

• Constraints:
  ▪ 24-hour time limit
  ▪ 1 job per node (cannot have multiple jobs on one node)
  ▪ Can set GPUs into Shared/Exclusive mode but not complete isolation
    (e.g., user that get access first can steal all the GPUs)
  ▪ Vendor specific with specific version of NVIDIA driver (>260)
Modeling max utilization

• With our approach using $n$ segments in 24-hour period:

$$
\text{utilization} = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - \sum_{i=1}^{n-1} \left( t_{\text{arrival}}(i) - t_{\text{lastchk}}(i) \right) + t_{\text{restart}}}{t_{\text{max}}} - \left( t_{\text{max}} - t_{\text{arrival}}(n) \right)
$$

• Without our approach:

$$
\text{utilization} = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - \left( t_{\text{arrival}}(1) - t_{\text{lastchk}}(1) \right) - \left( t_{\text{max}} - t_{\text{arrival}}(1) \right)}{t_{\text{max}}}
$$

where:

$$
t_{\text{arrival}}(i) = \begin{cases} 
    t_{\text{lastcheck}}(i) & \text{when } t_{\text{arrival}}(i) > t_{\text{max}} \\
    t_{\text{arrival}}(i) & \text{otherwise}
\end{cases}
$$

$$
t_{\text{lastchk}}(n) = f(\text{molecular system})
$$
Case study 1: Sodium Dodecyl Sulfate (SDS)

Initial structures: surfactant molecules randomly distributed

- Molar concentrations: 0.10
- Molar concentrations: 0.25
- Molar concentrations: 0.50
- Molar concentrations: 1.00
Case study 1: variable simulation times

![Graph showing performance (ns/day) vs. simulation time (ns) for different variable values.]
Case study 1: testbeds

• Taxonomy of our simulations:
  ▪ 4 concentrations and 3 200-ns trajectories per concentration at 298K
• Test 1:
  ▪ Jobs with same concentrations assigned to same node
• Test 2:
  ▪ Jobs with different concentrations assigned to same node
Case study 1: modeling max utilization

• With our approach using \( n \) segments in 24-hour period:

\[
utilization = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - \sum_{i=1}^{n-1} [t_{\text{restart}}] - (t_{\text{max}} - t_{\text{arrival}}(n))}{t_{\text{max}}}
\]

• Without our approach:

\[
utilization = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - (t_{\text{max}} - t_{\text{arrival}}(1))}{t_{\text{max}}}
\]

where:

\[
t_{\text{arrival}}(i) = \begin{cases} 
  t_{\text{lastcheck}}(i) & \text{when } t_{\text{arrival}}(i) > t_{\text{max}} \\
  t_{\text{arrival}}(i) & \text{otherwise}
\end{cases}
\]

\[
t_{\text{max}} = 24\text{hours}
\]
Case study 1: modeling arrival time

We model $t_{\text{arrival}}(i)$ in two ways:

- Scientists: run short simulation, compute ns/day, define job’s speed to constant rate to fit into 24-hour period
- Our approach: segment 24-hour job in segments, adjust segment length based on heuristic that takes into account change in ns/day
Case study 1: our heuristic

- Observed performance
- Projected performance
- Our heuristic
Case study 1: results

- Run 12 10-day trajectories with 4 concentrations and 3 different seeds on Keeneland, three trajectories per node

<table>
<thead>
<tr>
<th>$t_{chkpt}$ (hours)</th>
<th>With our approach</th>
<th>W/o our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>test 1</td>
<td>test 2</td>
</tr>
<tr>
<td>0.5</td>
<td>99.54%</td>
<td>98.82%</td>
</tr>
<tr>
<td>1</td>
<td>99.18%</td>
<td>98.44%</td>
</tr>
<tr>
<td>3</td>
<td>97.83%</td>
<td>96.98%</td>
</tr>
<tr>
<td>6</td>
<td>95.83%</td>
<td>94.85%</td>
</tr>
</tbody>
</table>
Case study 1: snapshots of ongoing simulations

Initial structures: surfactant molecules randomly distributed

- Molar concentrations: 0.10, time: 22ns
- Molar concentrations: 0.25, time: 20ns
- Molar concentrations: 0.50, time: 20ns
- Molar concentrations: 1.00, time: 15ns
Case study 2: Carbon Nanotubes

- Study nanotubes in aqueous solutions and electrolyte solutions
  - Different temperatures
  - Different separations
- Scientific metrics:
  - Potential of mean force
  - Effect of electrolytes, i.e., sodium chloride and iodide
  - Ion spatial distributions
Case study 2: testbeds

• Taxonomy of the simulations:
  ▪ 10 temperatures ranging from 280K to 360K along with 20 tube separations
  ▪ 200ns per trajectory with 5.8ns+-3% per day on 64 nodes

• Test 1:
  ▪ Hardware errors, i.e., ECC error and system failures

• Test 2:
  ▪ Hardware and application errors
Modeling max utilization

• With our approach:

\[
\text{utilization} = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - \sum_{i=1}^{n-1} \left( t_{\text{arrival}}(i) - t_{\text{lastchk}}(i) + t_{\text{restart}} \right) - (t_{\text{max}} - t_{\text{arrival}}(n))}{t_{\text{max}}}
\]

• Without our approach:

\[
\text{utilization} = \sum_{\text{days}} \sum_{\text{GPUs}} \frac{t_{\text{max}} - \left( t_{\text{arrival}}(1) - t_{\text{lastchk}}(1) \right) - (t_{\text{max}} - t_{\text{arrival}}(1))}{t_{\text{max}}}
\]

where: 
\[
t_{\text{arrival}}(i)_{i<n} = \text{weibull}(\text{scale,shape})
\]
\[
t_{\text{arrival}}(n) = 0.03 \times t_{\text{max}}
\]
\[
t_{\text{max}} = 24 \text{ hours}
\]
Case study 2: modeling system failures

• Weibul distribution: scale = 203.8 and shape = 0.525

P(system failure) = 0.057
P(two more jobs fail because of system given that one already failed) = 0.333
Case study 2: modeling application failures

- Weibul distribution: scale = 56.56, shape = 0.3361
Case study 2: results

- Run 200ns for each nanotube system – equivalent to ~35 days on 64 nodes of Keeneland, each with 3 GPUs

<table>
<thead>
<tr>
<th>$t_{chkpnt}$ (hours)</th>
<th>With our approach</th>
<th>W/o our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sysfail</td>
<td>sysfail</td>
</tr>
<tr>
<td>0.5</td>
<td>99.69%</td>
<td>99.54%</td>
</tr>
<tr>
<td>1</td>
<td>99.64%</td>
<td>99.47%</td>
</tr>
<tr>
<td>3</td>
<td>99.47%</td>
<td>99.23%</td>
</tr>
<tr>
<td>6</td>
<td>99.28%</td>
<td>98.98%</td>
</tr>
</tbody>
</table>
Case study 2: scientific results
Case study 2: scientific results
Case study 2: scientific results
Conclusions

• GPUs are still second class citizens on high-end clusters
  - Virtualization is too costly
  - Lightweight, user-level OSs are work in progress
• Rather than rewriting existing workflow and resource managers, we propose to complement them with:
  - Companion Module complementing the workflow manager
  - Wrapper Module supporting the resource managers
• We model the maximum usability for:
  - SDS systems with dynamically variable runtimes
  - Carbon nanotube systems with hardware and application failures
• Usability increases significantly in both cases
• The science is work in progress
  - Stay tune for our next publications
Acknowledgments

Related work:
- Taufer et al., CiSE 2012
- Ganesan et al., JCC 2011
- Bauer et al., JCC 2011
- Davis et al., BICoB 2009

Sponsors:

Contact: taufer@udel.edu, sapatel@udel.edu