Automated Load Balancing Invocation based on Application Characteristics

Harshitha Menon, Nikhil Jain, Gengbin Zheng, Laxmikant Kalé

25th September
Cluster 2012, Beijing, China
Outline

1. Introduction
   - Motivation
   - Load Balancing Challenges

2. Background

3. Meta-Balancer
   - Statistics Collection
   - Decision Making

4. Evaluation

5. Conclusion and Future Work
Outline

1. Introduction
   - Motivation
   - Load Balancing Challenges

2. Background

3. Meta-Balancer
   - Statistics Collection
   - Decision Making

4. Evaluation

5. Conclusion and Future Work
Motivation

- Modern parallel applications on large systems
  - Difficult to program and extract best performance
  - Performance is limited by most overloaded processor
  - The chance that one processor is severely overloaded gets higher as no of processors increases
Motivation

- Modern parallel applications on large systems
  - Difficult to program and extract best performance
  - Performance is limited by most overloaded processor
  - The chance that one processor is severely overloaded gets higher as no of processors increases

- Load imbalance in parallel applications
  - Leads to drop in system utilization
  - Hampers scalability of the application
Load Balancing Challenges

- Load balancing has to be profitable!
Load Balancing Challenges

- Load balancing has to be profitable!
- Determining factors
  - Incurred overheads - collection of statistics, execution of strategy to find the new mapping of tasks/work units, moving the tasks
  - When to perform load balance?
  - Load balancing strategy selection
Load Balancing Challenges

- Load balancing has to be profitable!
- Determining factors
  - Incurred overheads - collection of statistics, execution of strategy to find the new mapping of tasks/work units, moving the tasks
  - When to perform load balance?
  - Load balancing strategy selection
- Adaptive load balancing is needed in a dynamic applications
Meta-Balancer

- Automating load balancing related decision making
Meta-Balancer

- Automating load balancing related decision making
- Monitors the application continuously and predicts load behavior
Automating load balancing related decision making

Monitors the application continuously and predicts load behavior

Identifies when to invoke load balancing for optimal performance based on
- Predicted load behavior and guiding principles
- Performance in recent past
Outline

1. Introduction
   - Motivation
   - Load Balancing Challenges

2. Background

3. Meta-Balancer
   - Statistics Collection
   - Decision Making

4. Evaluation

5. Conclusion and Future Work
Meta-Balancer

Background

Charm++

- Message-driven parallel programming paradigm based on overdecomposition and migratable objects
Charm++

- Message-driven parallel programming paradigm based on overdecomposition and migratable objects
- Programmer decomposes the problem into tasks
- Charm++ RTS manages the scheduling of tasks on the processors
Charm++

- Message-driven parallel programming paradigm based on overdecomposition and migratable objects
- Programmer decomposes the problem into tasks
- Charm++ RTS manages the scheduling of tasks on the processors

System implementation

User View
Dynamic Load Balancing Framework in Charm++

- Based on principle of persistence
Dynamic Load Balancing Framework in Charm++

- Based on principle of persistence
- Instruments the application tasks at fine-grained level
Meta-Balancer

Dynamic Load Balancing Framework in Charm++

- Based on principle of persistence
- Instruments the application tasks at fine-grained level
- Relies on application user to invoke load balancer and select load balancing strategy
Dynamic Load Balancing Framework in Charm++

- Based on principle of persistence
- Instruments the application tasks at fine-grained level
- Relies on application user to invoke load balancer and select load balancing strategy
- When the load balancing is invoked
  - Gathers the statistics based on the strategy (centralized or hierarchical)
  - Executes load balancing strategy
  - Migrates objects based on new mapping
Outline

1. Introduction
   - Motivation
   - Load Balancing Challenges

2. Background

3. Meta-Balancer
   - Statistics Collection
   - Decision Making

4. Evaluation

5. Conclusion and Future Work
Design Overview

- Module to control load balancing related decision making
Design Overview

- Module to control load balancing related decision making
- Implemented on top of Charm++ load balancing framework
Design Overview

- Module to control load balancing related decision making
- Implemented on top of Charm++ load balancing framework
- Key responsibilities
  - Monitor the application: collect minimal statistics
  - Identify the iteration to invoke load balancing to optimize performance
  - Form a consensus among participating processors on when to invoke load balancing
Asynchronous collection
- Overlaps with application execution
- Supported using Charm++’s tree based reduction
- No barrier for statistics collection
Asynchronous collection
- Overlaps with application execution
- Supported using Charm++’s tree based reduction
- No barrier for statistics collection

Minimal statistics
- Max load
- Average load
- Utilization of processors
Consider the load imbalance given by

\[ \zeta = \frac{L_{\text{max}} - L_{\text{avg}}}{L_{\text{avg}}} \]
Consider the load imbalance given by

\[ \zeta = \frac{L_{\text{max}} - L_{\text{avg}}}{L_{\text{avg}}} \]

- \( \zeta > 0 \) means load imbalance; leads to performance loss
  - Should load balancing be invoked when \( \zeta > 0 \)?
Consider the load imbalance given by

\[ \zeta = \frac{L_{\text{max}} - L_{\text{avg}}}{L_{\text{avg}}} \]

- \( \zeta > 0 \) means load imbalance; leads to performance loss
  - Should load balancing be invoked when \( \zeta > 0 \)?
- Goal - minimize total execution time (application + load balancing overheads)
Model to Predict Ideal LB Period

- Consider a linear model for load prediction based on collected statistics
Model to Predict Ideal LB Period

- Consider a linear model for load prediction based on collected statistics
- Average load is represented by

\[ L_{avg} = a \times t + l_a \]
Model to Predict Ideal LB Period

- Consider a linear model for load prediction based on collected statistics.

- Average load is represented by

\[ L_{avg} = a \times t + l_a \]

- Max load is represented by

\[ L_{max} = m \times t + l_m \]
Model to Predict Ideal LB Period

Application execution time is sum of

- Time spent on running application
- Load Balancing overhead
Model to Predict Ideal LB Period

Application execution time is sum of

- Time spent on running application
- Load Balancing overhead

\[
\Gamma = \frac{\eta}{\tau} \times \left( \int_{0}^{\tau} (mt + l_m)dt + \Delta \right) + \int_{0}^{\eta} (at + l_a)dt
\]

- $\tau$ be the ideal LB period,
- $\eta$ be the total iterations an application executes,
- $\Gamma$ be the total application execution time, and
- $\Delta$ be the cost associated with load balancing
Equating the differential of total time to zero to minimize it, we obtain

$$\frac{d}{d\tau} (\Gamma) = \eta \times \left( \frac{m}{2} - \frac{\Delta}{\tau^2} \right) = 0$$

$$\tau = \sqrt{\frac{2\Delta}{m}}$$
Model to Predict Ideal LB Period

Equating the differential of total time to zero to minimize it, we obtain
\[
\frac{d}{d\tau} (\Gamma) = \eta \times \left( \frac{m}{2} - \frac{\Delta}{\tau^2} \right) = 0
\]
\[
\tau = \sqrt{\frac{2\Delta}{m}}
\]
Consensus Mechanism
Outline

1 Introduction
   - Motivation
   - Load Balancing Challenges

2 Background

3 Meta-Balancer
   - Statistics Collection
   - Decision Making

4 Evaluation

5 Conclusion and Future Work
Evaluation

Applications

- LeanMD: molecular dynamics simulation program
- Fractography: used to study fracture surfaces of materials
Evaluation

- **Applications**
  - LeanMD: molecular dynamics simulation program
  - Fractography: used to study fracture surfaces of materials

- **Machines used**
  - Ranger: SUN constellation cluster at TACC
  - Jaguar: Cray system at ORNL
Evaluation

- Applications
  - LeanMD: molecular dynamics simulation program
  - Fractography: used to study fracture surfaces of materials

- Machines used
  - Ranger: SUN constellation cluster at TACC
  - Jaguar: Cray system at ORNL

- Three sets of Experiments
  - No Load Balancing
  - Periodic Load Balancing
  - Using Meta-Balancer
LeanMD with No Load Balancing

- Overall processor utilization is 65%
- No significant variation in processor loads during the run
LeanMD with Periodic Load Balancing

- Frequent load balancing increases execution time
- Periodic load balancing may not give performance benefit
LeanMD with Meta-Balancer

- Invoked load balancer at the beginning
- Thereafter frequency of load balancing is low
- Improved performance by 31% and the overall utilization to 95%
LeanMD - Comparison of Execution Time

<table>
<thead>
<tr>
<th>Core</th>
<th>No LB (s)</th>
<th>Periodic LB (Period) (s)</th>
<th>Meta-Balancer (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1945.16</td>
<td>1451.30 (200)</td>
<td>1388.29</td>
</tr>
<tr>
<td>256</td>
<td>1005.22</td>
<td>750.11 (200)</td>
<td>695.55</td>
</tr>
<tr>
<td>512</td>
<td>516.47</td>
<td>393.30 (400)</td>
<td>355.85</td>
</tr>
<tr>
<td>1024</td>
<td>264.15</td>
<td>209.64 (400)</td>
<td>190.52</td>
</tr>
<tr>
<td>2048</td>
<td>135.92</td>
<td>116.69 (400)</td>
<td>94.33</td>
</tr>
<tr>
<td>4096</td>
<td>70.68</td>
<td>69.6 (700)</td>
<td>57.83</td>
</tr>
</tbody>
</table>

Meta-Balancer outperforms periodic load balancing
Fractography with No Load Balancing

- Large variation in processor utilization
- Low utilization leading to resource wastage
Fractography with Periodic Load Balancing

- Frequent load balancing leads to high overhead and no benefit
- Infrequent load balancing leads to load imbalance and results in no gains
Fractography with Meta-Balancer

- Identifies the need for frequent load balancing in the beginning
- Frequency of load balancing decreases as load becomes balanced
- Increases overall processor utilization and gives gain of 31%
Outline

1. Introduction
   - Motivation
   - Load Balancing Challenges

2. Background

3. Meta-Balancer
   - Statistics Collection
   - Decision Making

4. Evaluation

5. Conclusion and Future Work
Conclusion

- Difficult to find the optimum load balancing period
  - Depends on the application characteristics
  - Depends on the machine the application is run on
Conclusion

- Difficult to find the optimum load balancing period
  - Depends on the application characteristics
  - Depends on the machine the application is run on
- Meta-Balancer automates the decision of when to invoke load balancing based on application characteristics
Conclusion

- Difficult to find the optimum load balancing period
  - Depends on the application characteristics
  - Depends on the machine the application is run on
- Meta-Balancer automates the decision of when to invoke load balancing based on application characteristics
- Meta-Balancer adaptively identifies load balancing period
Conclusion

- Difficult to find the optimum load balancing period
  - Depends on the application characteristics
  - Depends on the machine the application is run on
- Meta-Balancer automates the decision of when to invoke load balancing based on application characteristics
- Meta-Balancer adaptively identifies load balancing period
- Meta-Balancer obtains substantial gains and avoids repetitive experimentation
Future Work

- Extend Meta-Balancer to select load balancing strategy
  - Computation vs Communication strategy
  - Refinement vs Comprehensive strategy
  - Centralized vs Distributed strategy
Future Work

- Extend Meta-Balancer to select load balancing strategy
  - Computation vs Communication strategy
  - Refinement vs Comprehensive strategy
  - Centralized vs Distributed strategy
- Better models for predicting load
Thank you!