1. INTRODUCTION

The dynamic behavior of large modern parallel simulation codes can lead to imbalances in computational load among processors. In this thesis, I address how to evaluate load imbalance at runtime and make its correction affordable.

2. QUANTIFYING THE EFFECTIVENESS OF LOAD BALANCING ALGORITHMS

- Statistical load metrics do not shed light on how to correct the imbalance.
- To evaluate and correct the imbalance, we need to attribute load to the migratory tasks in the application.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>(a)</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>2</td>
<td>50</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(c)</td>
<td>2</td>
<td>50</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(d)</td>
<td>2</td>
<td>50</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Need a model of:
- Weighted migratory tasks
- Relationships between tasks

When and how should we balance?
- Model costs associated with using different balancing methods
- Use model to select the load balancer method that achieves the lowest runtime

3. EFFICIENT LOAD BALANCE ALGORITHM FOR N-BODY SIMULATIONS WITH NON-UNIFORM DENSITY

Tasks of Highly Variable Sizes are Difficult to Balance

<table>
<thead>
<tr>
<th>Load Balancer Accuracy</th>
<th>Less Uniform Tasks</th>
<th>More Uniform Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processes</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>Process Size</td>
<td>2.048</td>
<td>2.048</td>
</tr>
</tbody>
</table>

![Graph showing load balancer accuracy for different processes and task sizes]

Our Method Uses Aggressive Adaptive Sampling to Define Uniform Tasks in Highly Non-Uniform Density

![Graph showing load balancer accuracy for different processes and task sizes]

4. LAZY LOAD BALANCING: OFFLOADING LOAD BALANCE COMPUTATION TO NON-APP. RESOURCES

Balance Asynchronously to the App-
- Run load balancer on a separate set of nodes in parallel to application computation
- Evaluate and decide how to correct imbalance without pausing the application
- Overlap LB method with application computation
- Decouple LB method partition size from app. partition size
- Effects of delayed decision?
- Impact of application drift?
- Impact of system scale?

Application and LB Algo scale differently → decouple resources
- LB algorithm performance varies with resources (i.e., graph partitioner)
- Communication overhead depends on resources reserved for LB algorithm

Run the load balance algorithm in a separate partition, asynchronously to the application

![Graph showing load balancer accuracy for different processes and task sizes]

5. CONTRIBUTIONS

- Lazy load balancing framework that decouples and offloads the load balance computation to make it affordable at scale
- A model for selecting the right load balance algorithm for the job
- An accurate and fast method to balance N-body applications with highly non-uniform density

![Graph showing load balancer accuracy for different processes and task sizes]