Asynchronous contact mechanics (ACM) is a method to simulate flexible material subject to complex collision and contact geometries, e.g., cloth simulation in animation. It uses nested penalty layers for collision responses. Asynchronous integration localizes the computation to regions where it is needed. We present a scalable parallel implementation of ACM using Charm++, a parallel runtime system.

Approach

We conduct broad phase collision detection in two levels:
1. Locally inside each partition, we use a 26-DOP hierarchy to fit the swept volumes of the triangles to detect potential collisions.
2. Globally among all the partitions, we fit the projection of each triangle to a 3D bounding box and then pass them to the existing collision detection library in Charm++.

### Broad Phase Collision Detection

- Visual-based parallel collision detection library in Charm++:
  - Filter the potential collision pairs based on their bounding boxes to quickly eliminate non-colliding pairs.
  - Minimize false positives due to conservative nature of bounding boxes.

### Narrow Phase Collision Detection

- Enhances load balance if more collisions are detected in the blue region.
  - Reduce the number of intra-node messages.
  - Achieve balance of the inter-node requests.

**Communication Imbalance**
- Over decompose work within node.
- Asynchronous message-driven processors with less communication can naturally offload the computation work from the processor with more communication work.
- Prioritized execution: communication request gets processed as soon as possible.

**Collision Response**
- A node-level phase barrier to delay the next long computation if necessary.
- Over decompose the penalty force calculation to achieve better load balance within a node.

### Parallelization Challenges

- Very fine-grained computation
  - The average computation grain size is 10 to 100 microseconds. The automated scheduling of the computation object in Charm++ helps us execute them more efficiently and overlap the computation and communication.

- Highly irregular communication pattern
  - The communication pattern is unpredictable and changes dynamically or stochastically due to collisions. A message-driven parallel runtime system like Charm++ fits naturally with the dynamic message pattern of ACM.

- Dynamic load balancing
  - Some more collisions in the blue region, and fewer in the red region. With the adaptive features of Charm++, we are able to achieve better load balance.

**Scalable Asynchronous Contact Mechanics with Charm++**

Introduction

In each step, forces are first applied optimistically multiple times in a collision window without attempting to find or resolve new collisions. Thereafter, collision detection is performed which may lead to a rollback if collisions are detected.

Overall Flow

Parallelization Challenges

Preliminary Results

The runtime for the Charm++ version of ACM is collected on Edison and Brickland. Edison is a Cray XC30 supercomputer at NERSC. Each node of Edison has two sockets, each socket is populated with a 12-core Intel “Ivy Bridge” processor E5-2695@2.4GHz. Brickland is a 4-socket system with Intel E7-4890@2.8 GHz (15 core Ivy Bridge). The TBB version of ACM is conducted on Brickland.

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Charm++ (Edison)</th>
<th>Charm++ (Brickland)</th>
<th>TBB (Brickland)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17035</td>
<td>23633</td>
<td>19805</td>
</tr>
<tr>
<td>3</td>
<td>52304</td>
<td>69473</td>
<td>59219</td>
</tr>
<tr>
<td>5</td>
<td>104625</td>
<td>138948</td>
<td>118432</td>
</tr>
<tr>
<td>8</td>
<td>199422</td>
<td>268013</td>
<td>236832</td>
</tr>
<tr>
<td>12</td>
<td>399582</td>
<td>537026</td>
<td>473664</td>
</tr>
<tr>
<td>24</td>
<td>799164</td>
<td>1074050</td>
<td>947328</td>
</tr>
<tr>
<td>48</td>
<td>1598299</td>
<td>2148104</td>
<td>1894656</td>
</tr>
<tr>
<td>96</td>
<td>3196598</td>
<td>4296208</td>
<td>3789312</td>
</tr>
<tr>
<td>192</td>
<td>6393196</td>
<td>8592416</td>
<td>7578624</td>
</tr>
<tr>
<td>384</td>
<td>12786392</td>
<td>17184832</td>
<td>15157248</td>
</tr>
<tr>
<td>768</td>
<td>25572784</td>
<td>34369664</td>
<td>30314496</td>
</tr>
</tbody>
</table>