Distributed Load Balancing Strategies with Charm++

Presented by
Simeng Liu, Kavitha Chandrasekar
Outlines

• Load Balancers Analysis
  • Prefix
  • Orthogonal recursive bisection (ORB)
  • Diffusion

• Application Characteristics
  • Iterative
  • Spatial locality with coordination
  • Well-scaled
Charm++ Load Balancing Infrastructure

Tool: migrate chare objects

Structures:
- TreeLB (2-4 levels)
Charm++ Load Balancing Infrastructure

Tool: migrate chare objects

Structures:
- TreeLB (2-4 levels)
- DistributedLB
  - each PE makes individual decisions
Prefix Based LB with ParaTreeT

Comparison of CentralLB and DistributedLB implementations:

Stampede2
SKX
48 cores / node
6144 cores on 128 nodes
Prefix Based LB with ParaTreeT

Composition of LB time:

<table>
<thead>
<tr>
<th></th>
<th>Initialization time</th>
<th>Strategy time</th>
<th>Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect object data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Measured</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>runtime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>graph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply LB strategy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and make migration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decisions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migrate objects</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prefix Based LB with ParaTreeT

Initialization time analysis:

Let

\[ P := \text{the number of PEs} \]
\[ O := \text{the number of objects} \]

<table>
<thead>
<tr>
<th>LB</th>
<th>Runtime</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrefixLB</td>
<td>( \log P )</td>
<td>O</td>
</tr>
<tr>
<td>DistributedPrefixLB</td>
<td>1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Reduction
Prefix Based LB with ParaTreeT

Strategy time analysis:

Let

\[ P := \text{the number of PEs} \]
\[ O := \text{the number of objects} \]

<table>
<thead>
<tr>
<th>LB</th>
<th>Runtime</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrefixLB</td>
<td>OlogO</td>
<td>N/A</td>
</tr>
<tr>
<td>Distributed PrefixLB</td>
<td>logP</td>
<td>log P</td>
</tr>
</tbody>
</table>

Sorting

Recursive doubling
Prefix Based LB with ParaTreeT

Migration time analysis:

Let

\[ P := \text{the number of PEs} \]
\[ O := \text{the number of objects} \]
\[ M := \text{number of objects need migration} \]

<table>
<thead>
<tr>
<th>LB</th>
<th>Runtime</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrefixLB</td>
<td>M/P</td>
<td>M</td>
</tr>
<tr>
<td>Distributed PrefixLB</td>
<td>M/P</td>
<td>M</td>
</tr>
</tbody>
</table>
ORB Based LB with ParaTreeT

Algorithm:
• Goal:
  - Partition the universe into number of PE blocks with even loads
• Centralized:
  - Use selection algorithm to find a splitting coordinate along the longest dimension of the subspace
• Distributed:
  - One partition: find a splitting coordinate
ORB Based LB with ParaTreeT

Algorithm:
- Distributed:
  - One partition: find a splitting coordinate

PE array

Bcast Subspace boundary coordinates

Check if the optimal bin splitter satisfy the tolerance

Reduce histogram binning of object loads Within the subspace

If yes, done

If no, divide the search space and repeat

May repeat too may times!
**Algorithm:**

- **Distributed:**
  - One partition: find a splitting coordinate

**ORB Based LB with ParaTreeT**

- Leader PE
  - Check if the optimal bin splitter satisfy the tolerance
  - If yes, done
  - If the optimal bin have less than THRESHOLD objects:
    - Collect coordinations of those objects directly

---

**PE array**

- Bcast
- Subspace boundary coordinates
- Reduce histogram binning of object loads Within the subspace
ORB Based LB with ParaTreeT

Algorithm:
• Distributed:
  - make number of PE partitions

PE array | First round | Second round | Third round
---------|-------------|--------------|-------------
0        |             |              |             |
1        |             |              |             |
2        |             |              |             |
3        |             |              |             |
4        |             |              |             |
5        |             |              |             |
6        |             |              |             |
7        |             |              |             |
ORB Based LB with ParaTreeT

Algorithm:
• Distributed:
  - make multiple reductions with different roots
Let

\[ P := \text{the number of PEs} \]

\[ V := \text{the number of objects} \]

**ORB Based LB with ParaTreeT**

Strategy time analysis: 9478 objects 768 cores 12

<table>
<thead>
<tr>
<th>LB</th>
<th>Runtime</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrbLB</td>
<td>VlogP</td>
<td>N/A</td>
</tr>
<tr>
<td>Distributed OrbLB</td>
<td>V(logP)^2</td>
<td>VlogP</td>
</tr>
</tbody>
</table>
Summary

• Analysis of the LB runtime with a three-stage decomposition

<table>
<thead>
<tr>
<th>Initialization time</th>
<th>Strategy time</th>
<th>Migration</th>
</tr>
</thead>
</table>

- Try the DistributedLB if strategy runtime $\leq \log(P)$

• Will Improve DistributedOrbLB
  - ORB to the node level
  - Diffusion within each node