The Graph Partitioning Problem

Given:
A graph \( G = (V, E) \) and number of partitions, \( p \)

Output:
A (vertex) partition \( V = V_0 \cup V_1 \cup \ldots \cup V_{p-1} \)

Such that:
1. \( \{V_i\} \) are disjoint \( \Rightarrow V_i \cap V_j = \emptyset \)
2. \( \{V_i\} \) are roughly balanced \( \Rightarrow |V_i| \approx |V_j| \)
3. Let \( E_{\text{cut}} = \{(u, v) | u \in V_i, v \in V_j, i \neq j\} \)

Minimize: \( |E_{\text{cut}}| \)

Multilevel Partitioning Algorithm:

Tasks in OpenMP

Independent units of work:
Composed of:
- Code to execute
- Data environment
- Internal control variables (ICV)
  - Threads perform the work of each task.
  - The runtime system decides when tasks are executed.
  - The task directive defines the code associated with the task and its data environment.

Reordering

Parallel x Serial Performance

The performance evaluations were run on 2 x Intel E5-2650 v3, 16 (2 x 8) cores with 64MB of DDR4 @ 2133MHz RAM. The graphs for the experiments were obtained at the University of Florida Sparse matrix collection (www.cise.ufl.edu). The graph names, sizes and execution times are described in the next panel.

As can be observed, speed ups of about 37 times on average were reached on our hardware set up. The speedup achieved is largely dependent upon the size of the graph and how well the graph is distributed among threads so each of them complete the task on their assignment list and start helping other threads.

<table>
<thead>
<tr>
<th>Name</th>
<th>Vtxs</th>
<th>Edges</th>
<th>Serial</th>
<th>2-Thr</th>
<th>8-Thr</th>
<th>16-Thr</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>448,695</td>
<td>3,314,611</td>
<td>5.79</td>
<td>2.30</td>
<td>0.62</td>
<td>0.33</td>
</tr>
<tr>
<td>m14b</td>
<td>214,765</td>
<td>1,679,018</td>
<td>2.66</td>
<td>0.98</td>
<td>0.25</td>
<td>0.17</td>
</tr>
<tr>
<td>wave</td>
<td>156,317</td>
<td>1,059,331</td>
<td>1.65</td>
<td>0.59</td>
<td>0.15</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 1 - Execution Times in Seconds

References