Evaluating the Efficiency of Parallel Large-scale Graph Processing Algorithms on Different Architectures

Ilya Afanasyev
Vadim Voevodin

Lomonosov Moscow State University
Motivation

- In modern supercomputing, a variety of different platforms, architectures, and configurations are used on the hardware side, paired with different compilers and programming models on the software side.

- With a variety of hardware and software choices, we have to analyse algorithm structure and properties using a co-design approach, to understand how different algorithms map on different architectures.

- We research the family of graph-processing algorithms, as large-scale graph processing problems are extremely relevant nowadays.
Graph algorithms co-design

- Problem (BFS, SSSP, etc)
- Input graphs (RMAT, SCCA2)
- Other input parameters
- Platform (GPU, Intel Xeon, Intel Xeon PHI, etc)
- Best algorithm?
- Best graph representation?
- Platform-dependent optimizations?

An efficient implementation for selected platform!
How to create an efficient implementation?

Information graph is a very powerful tool to study algorithm’s parallel properties!

Mathematical properties (complexity, required memory, required operations, data structures)

As a result of this analysis we get optimal algorithm - architecture pairs and with a detailed mathematical description for each algorithm.
Comparison to reference implementation

Selecting suitable algorithm and optimizing graph representation may lead to significant performance acceleration on its own!
So far we got some implementations capable of processing really large graphs on various architectures, but how can we prove that those implementations are actually effective?

<table>
<thead>
<tr>
<th>Задача</th>
<th>Intel Xeon CPU</th>
<th>NVidia GPU</th>
<th>Intel KNL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small graphs</td>
<td>Medium graphs</td>
<td>Large graphs</td>
</tr>
<tr>
<td>SSSP</td>
<td>170 MTeps</td>
<td>205 MTeps</td>
<td>121 MTeps</td>
</tr>
<tr>
<td>BFS</td>
<td>621 MTeps</td>
<td>986 MTeps</td>
<td>804 MTeps</td>
</tr>
<tr>
<td>SCC</td>
<td>63 MTeeps</td>
<td>110 MTeeps</td>
<td>89 MTeeps</td>
</tr>
<tr>
<td>Bridges</td>
<td>201 MTeeps</td>
<td>188 MTeeps</td>
<td>134 MTeeps</td>
</tr>
<tr>
<td>MST</td>
<td>86 MTeeps</td>
<td>85 MTeeps</td>
<td>53 MTeeps</td>
</tr>
<tr>
<td>Transitive closure</td>
<td>1295 MTeeps</td>
<td>1702 MTeeps</td>
<td>868 MTeeps</td>
</tr>
</tbody>
</table>
Evaluating the efficiency

How to understand if the selected algorithm is implemented efficiently for target architecture? We can:

- compare performance to some reference implementations (no implementations for some problems and architecture, no open-source code, etc)

- compare performance to other theoretical studies (no open-source code, different testing environment, different input data, modern architectures sometimes not covered well enough)

- develop a specific set of metrics to make sure we are reaching certain performance limit (right now using memory metrics since algorithms are memory-bound, future work - to extend this set)
Thank you for your attention!
Any questions?

Ilya Afanasyev
afanasiev_ilya@icloud.com