Heterogeneous many-core HPC architectures are the new norm.

Key hardware-specific features:

- (Haswell & Broadwell) and generic C++.

Backends

- Compartment loads & stores.
- CUDA

Work and memory requirements per node are proportional by a load balancer.

- Using MPI, TBB and C++11 threads;
- architectures;

- synapse models can be provided by simulator
- ffi

- Efficient cell state integration requires
- in the medium to long term.

- Fast key-value reduction for synapse cur-
- vectorised gather and scatter for per-

Arbor

Highlights of progress since the last HBP Summit:

- 512 nodes
- 256 nodes
- 64 nodes
- 1.91 x

- Progress & Features
- Resources consumption for 590k cells.
- Time to build model (20'600 cells/node)
- Validation tests against numeric/analytic models
- Reporting of memory and energy consumption
- Asynchronous spike exchange that overlaps
- AVX2.

- CARMA Algorithm:

[] repeat recursively

START

split the largest dim.

[] repeat recursively

P

P

P

P

processors

If not enough memory
execute sequentially

C = A × B

If enough memory
execute in parallel

P/4 P/4 P/4

communicate

Main Contributions:

CARMA before:

- Only powers of 2: \((m, n, k, P)\) assumed to be powers of 2 or that at each step the number of processor left and the largest dimension share common divisors.
- Cyclic data base-case layout: Requires complete data reshuffling after each communication. The corresponding mapper not provided. Compatibility issues with other layouts.
- Limited division schedules: Not all division schedules produced correct results.

CARMA now:

- Generalized implementation: Works for any \((m, n, k, P)\)
- Blocked data base-case layout: Requires local data reshuffling only in some case and only on the level of blocks, instead of single elements.
- Less memory, more performance: Buffers carefully allocated and reused throughout the algorithm. Using ~25% less memory in total.
- Any division strategy available

New Data Layout:

Cyclic Layout (before)

Requires local data reshuffling after each communication!

Buffer Reuse:

Our implementation decreases the total amount of memory used and the total number of buffers allocated. All the buffers are allocated just once and are then being reused throughout the application.

Conf (henceforth) in the plot means the configuration (e.g. 64 nodes, 8k cored per node)... which maximized the FPS was chosen.

Table 1:

<table>
<thead>
<tr>
<th>Problem size</th>
<th>square</th>
<th>two large dimensions</th>
<th>one large dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>m 64k 8k 64k 64k 64k 64k 8704 17408</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n 64k 8k 64k 8k 8k 8k 8704 17408</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k 64k 8k 8k 64k 8k 933888 3735552</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References:


Contact us:

source email | Marko Kabić (marko.kabic@cscs.ch)