Data optimization for linear algebra and stencil compact on Many-core processor

Etudiant : Minh Quan HO
Equipe : DRAKKAR
Directeur de thèse : Bernard TOURANCHEAU
Co-encadrant : Christian OBRECHT
Outline

- Context
- Applications
- Work done this year
- Conclusion and perspectives
- Exascale ($10^{18}$) by 2020
- Power $\leq 20$ MW
- $\geq 50$ Gflops/W

<table>
<thead>
<tr>
<th></th>
<th>Intel Xeon E5-2687W</th>
<th>Intel Xeon Phi co-processor 31S1P</th>
<th>Tesla K20X</th>
<th>Kalray MPPA-256</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gflops/W (FP64)</td>
<td>1.5</td>
<td>3.4</td>
<td>4.8</td>
<td>6.8</td>
<td>$\geq 50$</td>
</tr>
<tr>
<td>Number of cores/node</td>
<td>8</td>
<td>57</td>
<td>2688</td>
<td>256</td>
<td>?</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>3100</td>
<td>1100</td>
<td>732</td>
<td>400</td>
<td>?</td>
</tr>
</tbody>
</table>
## Applications

<table>
<thead>
<tr>
<th>Linear algebra (BLAS)</th>
<th>Compact stencils PDE solvers (e.g Lattice Boltzmann Method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Vector &amp; Matrix operations</td>
<td>• Access to neighbor nodes</td>
</tr>
<tr>
<td>• Irregular access pattern</td>
<td>• Non-contiguous memory access</td>
</tr>
</tbody>
</table>

- Cache coherence. How and Where ?
  - Hardware complexity & latency increases with # of cores
  - Incoherence-aware software design – Non-trivial
- Memory
  - Non-Uniform-Memory-Access (NUMA)
  - Difficult to do and maintain Global Memory + overhead
  - Bandwidth could hardly follow # of cores and clock
- Programming framework ?
1. Many-core as Stand-alone node
   - MPI design and implementation
   - MPI optimizations comparison

2. Many-core as Accelerator
   - clBLAS
   - clMAGMA

3. Benchmark
   - High Performance Linkpack (HPL)

Work done
1. Many-core as Stand-alone node

- Issues: Isolated cores, limited memory
- Design: 16 + 1 MPI processes per Many-core
  - MPI compute ranks: 16 x 2 MB on-chip memory
  - MPI I/O rank: 1 x 2 GB DDR
• Optimizations:
  • Eager send for short messages
  • Lazy send for medium messages

Lazy vs. Eager-splitting vs. DMA vs. Normal on MPI_Send. Between rank 0 and rank 15.
2. Many-core as Accelerator

CLBLAS

- Offloading computation from Host to Many-core
  - Static kernels for level 1-2
  - Dynamic code generation for level 3
- 2.6 Gflops (SGEMM) – 1.3 Gflops (DGEMM)

```c
do_GEMM(A, B, C,...);
do_something(){
  .
  .
  .
}
get_GEMM(C);
```
2. Many-core as Accelerator

CLMAGMA

- LAPACK-based interface (QR, LU, CHOL … )
- Auxiliary kernels written in OpenCL
- Use clBLAS to offload computation on device(s)
2. Many-core as Accelerator

CLBLAS / CLMAGMA issues

- Block decomposition too small compared to page-size
- Concurrent accesses to same memory block
- Kernels written for GPU, non-optimal for Many-core
3. Benchmark

- HPL benchmark with 16 MPI ranks, BLAS / OpenBLAS
- Limited local memory space (16 x 2 MB)
- Matrix size 1500 x 1500
- MPI Eager send gains 10% performance
Conclusion

- Many-core as Stand-alone node
  - MPI mono-Many-core, BLAS/OpenBLAS and HPL
  - Next:
    - MPI + DSM to increase memory space via I/O DDR + optimization modeling
    - MPI multi-Many-core for scalability
- Many-core as Accelerator
  - clBLAS + clMAGMA
  - Next:
    - Block decomposing tuning for Many-core
    - {Page|Cache}-size-aware kernels
    - LBM design & accelerating


Question ?