Review of

Kokkos Array Performance-Portable Manycore Programming model by

H. Carter Edwards and Daniel Sunderland

Erman Gurses
February 23, 2016
What problem did the paper address?

• Big picture problem
  • How can we provide performance portability to multi-core and many-core machines for scientific and engineering applications?

• Potential audience
  • Programming language,
  • Compiler, and
  • Parallel computing researchers

• Specific problem
  • Memory access patterns for different architectures have a big impact on performance.
Is it Important/Interesting?

- Already invested time for scientific and engineering applications

- Programming models always evolve

- Rewriting these applications from scratch is time consuming
What is the approach?

• The idea: Separating computational kernels from device-specific data access performance requirements

• Three main components
  • Many-core compute devices with its own memory
  • Data-parallel computational kernels
  • Multidimensional arrays

• Library-based programming model
What is the approach?

- Kokkos array API allows us to insert device-specific maps into the multi-dimensional array.
- The map determines device specific data access pattern.

Many-Core Compute Device

Data Parallel Computational Kernels

Multi-Dimensional Array Maps

Multi-Dimensional Array Data
How does the paper support or otherwise justify the conclusions it reaches?

- **Applications**
  - Explicit dynamics
  - Implicit thermal conduction
  - They are Mini Apps. using Finite Element Method

- **Device Specifications**

<table>
<thead>
<tr>
<th>Intel Westmere:</th>
<th>AMD Magny-Cours:</th>
<th>NVIDIA C2070:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon X5670 at 2.93 GHz</td>
<td>AMD Opteron 6136 at 2.4 GHz</td>
<td>NVIDIA C2070 at 1.2 GHz</td>
</tr>
<tr>
<td>Linux Kernel v2.6.18-194.el5</td>
<td>Linux Kernel v2.6.18-194.el5</td>
<td></td>
</tr>
<tr>
<td>24 pthreads on 2 cpus × 12 cores × 2 hyperthreads</td>
<td>16 pthreads on 2 cpus × 8 cores</td>
<td>compiled with CUDA v4 using -O3 -arch=sm_20</td>
</tr>
<tr>
<td>compiled with Intel v11 using -O3 optimization</td>
<td>compiled with Intel v11 using -O3 -arch=sm_20</td>
<td></td>
</tr>
</tbody>
</table>

- **Measurement**
  - Element Throughput

  \[
  \text{Total time required given operation} = \frac{\text{Element Throughput}}{\text{The number of elements}}
  \]

- **Given Operations**
  - Element Computations
  - Gather Assemble Steps

- **Programming Model**
  - Kokkos Array API

- **Plots**
  - Single and double precision results
  - X-Axis: the number of elements
  - Y-Axis: million elements per second
How does the paper support or otherwise justify the conclusions it reaches?

- They claim the results explains that Kokkos provides performance portability
- Explicit Dynamics Mini-Application
- Computationally dense

Explicit Dynamics Mini-Application in double precision

Explicit Dynamics Mini-Application in single precision
How does the paper support or otherwise justify the conclusions it reaches?

- They claim the results explain that Kokkos provides performance portability
- Implicit Thermal Conductivity Mini-Application
- Sparse linear fill is good when size is small (caching)
What problems are explicitly or implicitly left as future research questions?

• Explicit Questions
  
  • Why do not thread pinning improve the performance on AMD Magny-Cours device?
  • How can we provide knowledge of the index space in compile time?

• Implicit Questions
  
  • How can we provide shared memory utilization using Kokkos array API?
  • Do we have the performance portability for various types of applications?
  • Do we have performance portability for different kinds of devices?

• Anything else?
Critique

• Great punchline

• No explicit hypothesis

• “Global Memory” is used loosely
  • It is not clear when it is used for GPU or NUMA device

• How do we know the capabilities of device to reach the conclusion?
  • Stated “achieve performance which is commensurate with the capabilities of the device”

• One paragraph discussion for the results is not enough

• Why do not the demonstrate the results for CUDA hand coded applications? (Mentioned in Section 4.2)

• Device specifications can be more detailed.

• The paper does not mention shared memory at all, which is the one of most important feature in GPU computing

• Important contribution for performance portability
Relation to CSC 620

• With the advent of multi-core and many-core, what key features should a parallel programming model include?
  • NUMA devices, managing thread/memory
  • Many-core (GPU) coalesced memory access
• How can we separate implementation concerns from the algorithm specification?
  • Many-core compute devices with its own memory
  • Data-parallel computational kernels
  • Multidimensional arrays
• How should programming models be evaluated?
• Anything else?