Maximizing TTI RTM Throughput With Kepler

David Wade, Kristian Bendiksen & Tor Erik Rabben
TTI RTM

• RTM – Forward propagate source wavefield, back propagate receiver wavefield, correlate to generate image.

• TTI – Propagation velocity model has 5 parameters: v, ε, δ, θ, φ.

• We model wave propagation using discretisation of these equations by Fletcher et al. (2009):

\[
\frac{\partial^2 p}{\partial t^2} = v^2_{px} H_2 p + \alpha v^2_{pz} H_1 q + v^2_{sz} H_1 (p - \alpha q)
\]

\[
\frac{\partial^2 q}{\partial t^2} = \frac{v^2_{pn}}{\alpha} H_2 p + \alpha v^2_{pz} H_1 q + v^2_{sz} H_1 \left(\frac{1}{\alpha} p - q\right)
\]

Optimization challenging because:

- High utilisation of memory bandwidth is required.
- Large stencil sizes require huge numbers of floating point operations.

Does Kepler provide continued acceleration?
EAGE 2013 Summary

• We created a **CPU+GPU hybrid** version targeted at:
  - Dual socket Intel X5675 (Westmere)
  - Plus 2x nVidia M2090 (Fermi)

• The collaboration between Statoil and IBM demonstrated up to a **4.5x speedup** on a full production TTI RTM code over CPU only

• We proposed to evaluate the possibility of obtaining similar performance gains when comparing nVidia Kepler with Intel Sandybridge
Advantages of GPUs

**CPU**

- Fewer, more complex cores
- Memory access heavily dependant on caching
- Lower theoretical peak: 332 GFlop/s per node for dual socket E5-2670 (Sandybridge)

... but ...

- Relatively easy to program and obtain good performance.

**GPU**

- Very large number of simple cores
- High memory bandwidth (320 GB/s for K10)
- Massive theoretical peak: 9160 GFlop/s per node for 2x K10 (Kepler)

... but ...

- Difficult to program to even reach a fraction of peak performance.
Original CPU Design

- **Z Kernel – Internal Regions**: Compute partial 1st and 2nd order spatial derivatives of the P-wave and Q-wave w.r.t. z-dimension. (Green)

- **Z Kernel – Overlap Regions**: Maintain continuity of wave propagation across multiple grid zones, by using an adapted Z kernel for the boundaries between them. (Purple)

- **XY Kernel**: Update to P and Q-fields for time $t+1$ made using iterations $t$ and $t-1$ plus H1 and H2 along with multiple velocity and dip/azimuth angle parameters. (Purple and Green)

- **PML Kernel**: Handles the absorption of wavefield energy at 3D volume boundaries for the given 3D volume. (Orange)
We distribute the grids between the GPUs on the node, including memory for \( P(t+1), Q(t+1), P(t), Q(t), P(t-1), Q(t-1), v, \epsilon, \delta, \theta, \varphi \). (Max 16GB for 2xK10)

- **GPU - GPU** communication required for two grid zone boundary calculations per GPU

- **CPU - GPU** transfer restricted to:
  a) Initial velocity model copy
  b) Source/receiver insert

- **GPU - CPU** transfer *only* needed to offload every \( k^{th} \) P-Field for image correlation

- Utilizing both **CPU** and **GPU** during receiver loop greatly improves throughput
## Optimization details

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Z-Internal</th>
<th>Z-Overlap</th>
<th>XY</th>
<th>PML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loop Unrolling:</strong> Increased effective occupancy by improving ILP with loop unrolling during the calculation of derivatives, this reduces stall time due to memory reads from the global memory on the GPU.</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
</tr>
<tr>
<td><strong>Read-Only Cache:</strong> Coefficients for partial derivative calculation are common to all SMs and hence can be stored on the read-only cache.</td>
<td>🟢</td>
<td>🔴</td>
<td>🟢</td>
<td>🔴</td>
</tr>
<tr>
<td><strong>Shared Memory Usage:</strong> While shared memory has much lower latency than global memory, care must be taken to avoid using so much that not enough threadblocks can be launched.</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🔴</td>
</tr>
</tbody>
</table>
Choice of K10

- K20X provides more realisable compute power per GPU
- However, single precision performance per card on this problem is higher with K10
End-to-End Performance Results

• We compare the CPU ➔ Hybrid speedup for a range of problem sizes
• Original Fermi port:
  4x – 4.5x Westmere
• Kepler update:
  3x – 4x Sandybridge

Faster, but not as much as we want. Why?
Concurrency Limitation

- When work cannot be distributed evenly across GPUs some must waste time waiting.

- Going from 2 to 4 GPUs exacerbates the problem of grids being unequal.

- In this real example, utilization is:
  - 97% with 2 GPUs
  - 78% with 4 GPUs

  i.e. only ~3x speedup with 4x the compute resources!
Extra Kernel Performance

- Current implementation uses 2-pass approach from CPU design:
  - Z-kernel (internal and overlap): $\partial^2z, \partial z^2$
  - XY-kernel: $\partial^2x, \partial^2y, \partial^2z, \partial^2x^2, \partial^2y^2, \partial^2yz, \partial^2x^2y, \partial^2x^2z$

- This implementation makes extensive use of shared memory, however registers provide even lower latency

- Example code from Paulius Micikevicius has demonstrated that our XY-kernel could be successfully split in two:
  - $\partial^2yz$
  - $\partial^2x, \partial^2y, \partial^2x^2, \partial^2y^2, \partial^2x^2y, \partial^2x^2z$

- The resulting 3-pass approach:
  - Uses register queues in place of shared memory
  - Delivers a 2x speedup for a regular grid
  - Precludes us from leveraging varying XY spacing
Getting the most from K10

• Concurrency limitation only likely to get worse, as 6-GPU nodes become available:
  - Load-balanced GPU solution needed

• Cannot afford to throw away GPU cycles waiting for shared-memory latency:
  - Method to reduce compute load without grids needed

• Implication: To write a top-performing GPU code, a ground-up solution must be designed for GPU
  ➔ Proposed as future work
Summary and further work

• **TTI RTM** is a highly compute intensive and complex problem

• We have evolved our **CPU+GPU hybrid** version targeted at
  - 2x nVidia K10 (Kepler) = 4 GPUs
  - Dual socket Intel E5-2670 (Sandybridge) = 16 cores

• This new hybrid version demonstrates up to a **3.5x end-to-end speedup** on a full production TTI RTM code over Sandybridge CPU only, enough to make a cost-benefit case.

• However, we see limitations within the overarching design which lead us to the conclusion that a **ground-up GPU solution** is the way forward.
Maximizing TTI RTM Throughput With Kepler

David Wade
dawad@statoil.com
www.statoil.com

Thanks go to the collaborators on this project:

David Wade
Kristian Bendiksen
Tor Erik Rabben