© 2011 Schlumberger. All rights reserved.

An asterisk is used throughout this presentation to denote a mark of Schlumberger. Other company, product, and service names are the properties of their respective owners.
Schlumberger Abingdon Tech. Center

Based 8 miles from Oxford, UK

- ~200 employees involved in developing oilfield software
- ~50 people involved in commercial simulator development
  - ECLIPSE*: Established FORTRAN/MPI code focusing on high end physics
  - INTERSECT*: New to market C++/MPI code focusing on scalability and large model workflows
GPU Technology Evaluation

Have been evaluating GPU technology for the past 2+ years, particularly the NVIDIA CUDA architecture

1. How can GPUs be used in existing commercial products
2. Considerations for algorithm design
3. Considerations for software engineering

Innovate. For the Long Run.
Simulator Overview

- Evolutionary, timesteps from an initial state
- Solves a set of non-linear conservation equations
- Solved implicitly due to fast propagating pressure transients
- Newton iteration requires solution of linear system
- Solution of linear system is critical to performance but not naturally parallel
- Models routinely have $10^4 - 10^7$ grid blocks
Timestep Loop to Find Solution $x$ at $t + \Delta t$

1) Property calculations from $x$

2) Residual and Jacobian assembly

$$r_{comp}(x) = \frac{\partial c_{comp}}{\partial t}(x) - f_{comp}(x) - w_{comp}(x)$$

Accumulation \quad Flux \quad Source

3) Linear solve to find solution increment

$$\frac{\partial r}{\partial x} \delta x = r, \quad x = x - \delta x$$
### Suitable Components to Migrate to GPU

<table>
<thead>
<tr>
<th>Component</th>
<th>% run time &amp; % code base</th>
<th>Type of algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property calculations</td>
<td>15-30% run time 15+15% code base Ratio ~ 1</td>
<td>Largely independent cell calculations for fluid properties. Also includes well model. Many branches depending on model</td>
</tr>
<tr>
<td>Matrix Assembly</td>
<td>15 – 20% run time 1% code base Ratio ~ 15</td>
<td>Independent assembly of governing equations for each cell</td>
</tr>
<tr>
<td>Linear Solver</td>
<td>50 – 70% run time 2% code base Ratio ~ 30</td>
<td>Most potential but also the most sequential part of the code</td>
</tr>
</tbody>
</table>
GPU Linear Solver Components

CUDA memory bandwidth

- Product
- Factor
- Solve
- Guide

CUDA FP performance

- Product
- Factor
- Solve
- Guide
Should We Write a GPU Linear Solver?

OpenMP vs CUDA

Milliseconds

Number of threads

Intel Xeon X5482 & NVIDIA Tesla C1060

Innovate. For the Long Run.
Potential Demonstrated

- Ideas evolved into the Massively Parallel Nested Factorization (MPNF) algorithm
- Presented at the 2011 SPE RSS
- Results generated on an Intel Xeon X5550 with a NVIDIA Tesla C2050
Iterative Linear Solver Essentials

Solve $Mx = r$, constructing the solution in terms of a small number of basis vectors ($v$)

$$\tilde{x} = \sum \alpha_i v_i \quad \|M\tilde{x} - r\| < \varepsilon$$

Performance depends on the ability to generate good $v$’s

Preconditioner $N$ is an easy to invert approximation to $M$

Error in $x$ \(\xrightarrow{M} \) Error in $r$

\[ N^{-1} \]

$N$ often a factorization (LU) which is essentially a serial algorithm
A GPU Based Preconditioner

Color order grid so that columns are independent

All columns in a color can be factored in parallel. Solve $Ny=r$ as

$$
\begin{bmatrix}
A & s_R \\
C & s_B
\end{bmatrix}
\begin{bmatrix}
r_R \\
r_B
\end{bmatrix}
=\begin{bmatrix}
I & A^{-1}B \\
I - E
\end{bmatrix}
\begin{bmatrix}
y_R \\
y_B
\end{bmatrix}
=\begin{bmatrix}
s_R \\
s_B
\end{bmatrix}
$$
GPU Linear Solver Performance

Compared with serial CPU solver on a suite of 30 models with up to 900k grid blocks

- All models ran to completion with comparable results
- 50% more linear iterations
- 10% non-linear iterations & 10% more timesteps
- Due to trade off between parallelization and accuracy of preconditioner.
Speed-up for Linear Solver Only

Want to be operating here but GPU memory is a constraint
Overall Speed-up with GPU Linear Solver

Number of grid blocks vs. Overall Speed-up
Next Step – Assembly on the GPU

Graphs showing the relationship between the number of grid blocks and assembly speed-up and overall speed-up for GPU Solver and GPU Solver and assembly.
GPU Technology Evaluation Conclusions

1. How can GPUs be used in existing commercial products
   - With modest effort we can achieve a \(~3\times\) speed-up

2. Considerations for algorithm design

3. Considerations for development/architecture frameworks
Algorithm Design - Linear Solver Scalability (I)

Problem size fixed, increase number of GPUs

MPNF is algorithmically unchanged by a domain decomposition layer:

- Process same color simultaneously across all domains
- Corrections between colors are passed both within and between domains
- Same iteration count whether on one machine, or a cluster
- Under exploration
Algorithm Design - Linear Solver Scalability (II)

Problem size increasing, fixed number of GPUs

- MPNF has shown good scalability but GPU memory is a constraint on testing.
- MPNF is not as recursive as a multi-grid algorithm.
- SLB currently working on massively parallel recursive algorithms that might exhibit better scalability on very large problems.
Software Engineering – Past Experiences

- Scientific code can have a long life time
  - Clients still run old versions & models
- Working with non-proprietary standards has enabled ECLIPSE* to adapt to:
  - OS changes
  - Hardware changes
  - Interconnect changes
Software Engineering - Past Experiences

- Simulator developers are not generally Computer Scientists so need an effective environment where performance comes naturally
- Developers swap projects – need a productive environment

Continue to explore pros/cons of other solutions:

- Alternative hardware targets - Intel ArBB
- Platform neutral approaches - Oxford Parallel Library (OP2)
- Higher level implementations - CUDA Thrust
GPU Technology Evaluation Conclusions

1. How can GPUs be used in existing commercial products
   - With modest effort we can achieve a ~3x speed-up

2. Considerations for algorithm design
   - Have demonstrated that existing algorithms can be adapted. Massively parallel algorithms still an area of internal research.

3. Considerations for software engineering
   - Not yet obvious - but we know the cores are coming!

Innovate. For the Long Run.