Multi-GPU Implementation of LU Factorization

Yulu Jia  Piotr Luszczek  Jack Dongarra

ICL Lunch talk

2/24/2012
Overview of the talk

- Block LU algorithm
- Multi-core Multi-GPU LU factorization
- Verifying the numerical result
- Implementation
- Performance evaluation
Slove the equation:

\[ Ax = b \]
Motivation

Slove the equation:

\[ Ax = b \]

One way of solving it:

\[ PA = LU \]
Block LU Algorithm

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
= P
\begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
U_{11} & U_{12} \\
0 & U_{22}
\end{bmatrix}
= P
\begin{bmatrix}
L_{11}U_{11} & L_{11}U_{12} \\
L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22}
\end{bmatrix}
\]

\[U_{12} \leftarrow L_{11}^{-1}A_{12} \tilde{A}_{22} \]

\[A_{22} \leftarrow A_{22} - L_{21}U_{12} \]

\[L_{22}U_{22} \]

Yulu Jia, Piotr Luszczek, Jack Dongarra (UTK) Multi-GPU Implementation of LU Factorization ICL Feb 2012 4 / 26
Block LU Algorithm

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix} = P \begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix} \begin{bmatrix}
U_{11} & U_{12} \\
0 & U_{22}
\end{bmatrix} = P \begin{bmatrix}
L_{11}U_{11} & L_{11}U_{12} \\
L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22}
\end{bmatrix}
\]

which yields:

\[
U_{12} \leftarrow L_{11}^{-1} A_{12} \\
\tilde{A}_{22} \leftarrow A_{22} - L_{21}U_{12} = L_{22}U_{22}.
\]
Block LU step by step
Block LU step by step
Block LU step by step
Block LU step by step
Block LU step by step

\[
\begin{align*}
U_{11} & \quad \text{getrf} \quad L_{21} \\
& \quad \text{trsm} \quad U_{12} \\
& \quad \text{gemm} \quad \tilde{A}_{22}
\end{align*}
\]
Block LU step by step
Previous Implementations

- LINPACK pdgesv
- LAPACK dgetrf
- ScaLAPACK pdgetrf
- PLASMA dgetrf
- MAGMA dgetrf
A Multi-CPU Multi-GPU Platform
Hybrid LU Factorization

- CPUs exhibit reasonable performance for a wide range of instruction mixes
- GPUs are especially good at floating point operations on large data sets without branches
- Make use of both to combine the advantages
Main Strategy

- Use a few CPU cores to factorize the panel
- Use GPUs to update the trailing submatrix
- Use all the rest of the CPU cores to help to update the trailing submatrix
Data Distribution

- The host keeps a copy of the whole matrix
- Part of the matrix is replicated in the GPU memories
Data Layout

- Column major in host memory conforming to LAPACK format
- Row major in GPU memory to improve the pivoting efficiency
During the factorization we need to send data back and forth between the host and devices. The data link between the host and devices is slow compared with the GPU’s computation power. Solution: hide data movement by overlapping communication with GPU computation.
Algorithm 1 GPU algorithm

for every iteration do
  for every block column resident on the GPU and to the right of the current panel do
    call DTRSM to calculate part of $U_{12}$
    call DGEMM to calculate part of $\tilde{A}_{22}$
    if this block is the next panel then
      initiate the asynchronous data transfer
    end if
  end for
end for
Look-ahead is the reordering of execution. It eliminates the idle time of GPUs while they are waiting for the panel factorization result which is done by the CPU cores. This actually hides the panel factorization.
Parallel Panel Factorization

The panel factorization lies on the critical path, its performance affects the entire LU factorization. Parallelizing the panel factorization alleviates this drag.

Performance of panel LU factorization panel width set to 256.
Scheduling

- Static scheduling between CPU cores and GPUs
- Dynamic scheduling among CPU cores
Static Scheduling Between CPU Cores And GPUs
Dynamic Scheduling Among CPU Cores

- Divide core column blocks into square blocks
- A ready task queue for each column block owned by the CPUs
- A group of CPU cores are assigned to each core column block
Verifying the Numerical Result

1. Generate input matrix $A$, right hand side $b$
2. Solve $Ax = b$ using our factorization routine and LAPACK DGETRS

The result is validated with this formula:

$$\frac{\|Ax - b\|_\infty}{\|A\|_\infty \|x\|_\infty n\epsilon} \leq O(1)$$
Implementation

- **Programming language:** C
- **Threading:** Pthread
- **GPU math library:** MAGMA and CUBLAS 3.2
- **Blocking factor:** NB = 256
- **Synchronizing the panel factorization and the update:** volatile variables in C
- **Synchronizing CPU cores for the update:** counting semaphores
## Test Platform

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor model</td>
<td>AMD Opteron 6172</td>
<td>NVIDIA Tesla S2050</td>
</tr>
<tr>
<td>Clock frequency</td>
<td>2.1 GHz</td>
<td>1.15 GHz</td>
</tr>
<tr>
<td>Number of sockets</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Cores per socket</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>GPUs per socket</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Memory</td>
<td>124 GiB</td>
<td>3 GiB per GPU</td>
</tr>
<tr>
<td>Peak DP</td>
<td>8.4 Gflop/s</td>
<td>515 Gflop/s</td>
</tr>
<tr>
<td>Max DGEMM</td>
<td>7.59 Gflop/s</td>
<td>298.62 Gflop/s</td>
</tr>
<tr>
<td>BLAS/LAPACK</td>
<td>Intel MKL 10.3</td>
<td>CUBLAS 3.2</td>
</tr>
<tr>
<td>OS</td>
<td>Red Hat 4.1.2-48</td>
<td>-</td>
</tr>
<tr>
<td>Compiler</td>
<td>gcc version 4.1.2</td>
<td>nvcc V0.2.1221</td>
</tr>
<tr>
<td>System interface</td>
<td>Socket G34</td>
<td>PCIe x16 Gen2</td>
</tr>
</tbody>
</table>

Detailed specification of the experimental environment.
Performance Evaluation

The beginning of the execution trace

The middle part of the execution trace

The end of the execution trace

Execution trace of the algorithm
Performances of the algorithm with all the cores
Performance Evaluation

**Weak scalability**

**Strong scalability**

Scalability

Yulu Jia, Piotr Luszczek, Jack Dongarra (UTK)  Multi-GPU Implementation of LU Factorization  ICL Feb 2012
Efficiency

Peak performance of the system:
48 CPU cores + 4 GPUs = 2463.2 Gflop/s
Peak DGEMM performance of the system:
1558.8 Gflops/s
Our implementation reaches: 1134.56 Gflop/s, which is 46.06% of the theoretical peak, and 72.78% of the DGEMM peak.
The End