MAGMA: toward fast Eigensolver

Azzam Haidar

ICL Friday talk, November 30, 2012
General Overview: the Eigenproblem algorithms

- **Symmetric EVP** $\mathbf{Ax} = \lambda \mathbf{x}$ meaning compute $\mathbf{A} = \mathbf{Z} \lambda \mathbf{Z}^*$ where $\lambda$ are the eigenvalues and $\mathbf{Z}$ are the eigenvectors.

1. **Tri-Diagonalization Reduction:** transform $\mathbf{A}$ to nice form $\mathbf{A} = \mathbf{Q} \mathbf{T} \mathbf{Q}^*$

2. **Solve:** compute the Eigenvalue and Eigenvectors of the tridiagonal $\mathbf{T} = \mathbf{E} \lambda \mathbf{E}^*$

3. **Back transformation:** update the computed Eigenvectors. $\mathbf{Z} = \mathbf{Q}^* \mathbf{E}$
There are **two paths** to tridiagonal form

1. The standard LAPACK algorithm.

   Christian Bischof, Bruno Lang, Xiaobai Sun (94) proposed multiple-stage implementation called Successive Band Reductions to reduce a matrix to tridiagonal.
The standard Tridiagonal reduction xSYTRD

**Characteristics**

- Too many Blas-2 op,
- Relies on panel factorization,
- Total cost $4n^3/3$,
- **Bulk sync phases,**
- **Memory bound algorithm.**
The MAGMA full reduction to tridiagonal

Performance comparison standard approach 1-stage

flops formula: 
\[ 4n^3 / 3 \times \text{time} \]
Higher is faster

Keeneland
GPU M2090 (@1.3 GHz, peak 583 GFlop/s)
CPU Intel Xeon X5660@2.80GHz (2 x 6 cores)
Idea:

- The idea is to cast expensive memory operations, occurring during the panel factorization into fast compute intensive ones.
- Redesign the algorithm in a new fashion which increase the cache reuse.
- Design new cache friendly kernels to overcomes the memory bound limitation.
- Extract parallelism and schedule task in an asynchronous order.
The MAGMA reduction: 2 stage algorithm

**Characteristics**

- **Stage 1:**
  - BLAS-3,
  - one shot reduction,
  - asynchronous execution,

- **Stage 2:**
  - BLAS-1.5,
  - element-wise/column-wise,
  - asynchronous execution,
  - new cache friendly kernel.

The reduction:

CPU: QR on panel \( i \)

CPU: waiting next panel \( i+1 \)

GPU: compute \( W(i) \) and update next panel \( (i+1) \)

GPU: update trailing matrix

0: \( NT= \) number of block

1: for \( i = 0, 1 \) to \( NT-1 \)

2: \( \text{panel}_{step_i}: QR(i) \Rightarrow (V,T) \)

3: compute \( X = AVT \) (SYMM)

4: compute \( W = X - \left( \frac{1}{2} \right) V^*T^*V^*X \)

5: update trailing matrix
   \( A = A - W^*V - V^*W \) (SYR2K)

6: end for
MAGMA: first stage, reduction to band

The reduction:

- **GPU:**
  - Compute $W(i)$
  - Update trailing matrix $W(i+1)$

- **CPUs:**
  - QR on panel $(i+1)$

- **Step i**

Panel of $(i+1)$

GPU: compute $W(i)$ and update next panel $(i+1)$

CPU: QR on panel $(i+1)$

GPU: update trailing matrix
MAGMA: toward fast Eigensolver

flops formula: $\frac{4n^3}{3} \times \text{time}$

Keeneland
GPU M2090 (@1.3 GHz, peak 583 GFlop/s)
CPU Intel Xeon X5660@2.80GHz (2 x 6 cores)

Characteristics
- Too many Blas-2 op,
- Relies on panel factorization,
- Bulk sync phases,
- Memory bound algorithm.

Higher is faster
**MAGMA: toward fast Eigensolver**

- **flops formula:** \(4n^3/3 \times \text{time}\)
- **Higher is faster**

Keeneland
- **GPU** M2090 (@1.3 GHz, peak 583 GFlop/s)
- **CPU** Intel Xeon X5660@2.80GHz (2 x 6 cores)

**Characteristics**
- **Blas-2 GEMV moved to the GPU,**
- **Accelerate the algorithm by doing all BLAS-3 on GPU,**
- **Bulk sync phases,**
- **Memory bound algorithm.**
MAGMA: toward fast Eigensolver

![Graph showing performance of different algorithms]

flops formula: \( 4n^3/3 \times \text{time} \)
Higher is faster

Keeneland
GPU M2090 (@1.3 GHz, peak 583 GFlop/s)
CPU Intel Xeon X5660@2.80GHz (2 x 6 cores)

**Characteristics**
- **Stage 1**: BLAS-3, increasing computational intensity,
- **Stage 2**: BLAS-1.5, new cache friendly kernel,
- **4X/12X** faster than standard approach,
- **Bottleneck**: if all Eigenvectors are required, it has 1 back transformation extra cost.
MAGMA: toward fast Eigensolver

What about a multi GPU?
The reduction:

CPU: QR on panel $i$

CPU: waiting next panel $i+1$

GPU: compute $W(i)$ and update next panel ($i+1$)

GPU: update trailing matrix

0: $NT=\text{number of block}$
1: for $i = 0; 1 \text{ to } NT-1$
2: \hspace{1cm} panel_{step \ i} : QR(i) \Rightarrow (V,T)$
3: \hspace{1cm} compute $X = AVT$ (SYMM)
4: \hspace{1cm} compute $W = X - (\frac{1}{2})V^T V^* X$
5: \hspace{1cm} update trailing matrix $A = A - W^* V - V^* W$ (SYR2K)
6: end for
The reduction:

CPU: QR on panel \( i \)

CPU: waiting next panel \( i+1 \)

GPU: compute \( W(i) \) and update next panel \( (i+1) \)

GPU: update trailing matrix

---

0: NT= number of block
1: for \( i = 0; 1 \) to \( NT-1 \)
2: panel-step: \( QR(i) \rightarrow (V,T) \)
3: compute \( X = AVT \) (SYMM)
4: compute \( W = X - \left( \frac{1}{2} \right) V^T V^* X \)
5: update trailing matrix \( A = A - W^* V - V^* W \) (SYR2K)
6: end for
MAGMA: first stage, reduction to band

The reduction: ZHEMM

\[
\begin{bmatrix}
0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\end{bmatrix} \times \begin{bmatrix}
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix}
\]
MAGMA: first stage, reduction to band

The reduction: \[ \text{ZHEMM way 1: } C(i) = A(i:n,i)' \times B(i:n) \]

Diagram: 

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

bad perf
MAGMA: first stage, reduction to band

The reduction: 

\[ \text{ZHEMM way1: } C(i) = C(i) + A(i, \text{myblk}) \times B(\text{myblk}) \]
MAGMA: first stage, reduction to band

The reduction: ZHEMM way2: \[ C(i:n) = A(i:n,i) \times B(i) \]
MAGMA: first stage, reduction to band

The reduction: \[ \text{ZHEMM way2: } C(\text{myblk}) = A(i,\text{myblk})' \times B(i) \]

Better perf
MAGMA: first stage, reduction to band

The reduction: ZHEMM way2

need sum allreduce

x =
MAGMA: first stage, reduction to band

The reduction: \texttt{ZHEMM DSYMM}

- GPU0
- GPU1
- GPU2
- GPU3
- GPU4
- GPU5
- GPU6
- GPU7

Sending to CPU: Red arrows
Receiving from CPU: Blue arrows
MAGMA: first stage, reduction to band

The reduction:

sending to CPU

comm

ZHEMM

ZHER2K

receiv from CPU
MAGMA: first stage, reduction to band

The reduction: ZHETRD_HE2HB (time sec)

<table>
<thead>
<tr>
<th></th>
<th>1GPU</th>
<th>2GPU</th>
<th>3GPU</th>
<th>4GPU</th>
<th>5GPU</th>
<th>6GPU</th>
<th>7GPU</th>
<th>8GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>18</td>
<td>15</td>
<td>13.6</td>
<td>13.9</td>
<td>14.3</td>
<td>14.9</td>
<td>15.8</td>
<td>17.1</td>
</tr>
<tr>
<td>v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>61</td>
<td>43</td>
<td>36.7</td>
<td>34</td>
<td>33.2</td>
<td>33.5</td>
<td>34.4</td>
<td>36.8</td>
</tr>
<tr>
<td>v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MAGMA: first stage, reduction to band

The reduction: \texttt{ZHEMM DSYMM}

Diagram showing the process with GPUs and CPU.
MAGMA: first stage, reduction to band

The reduction: **ZHEMM DSYMM**

![Diagram showing the reduction process involving GPUs and CPU]

- **GPU0** to **GPU1**
- **GPU2** to **GPU3**
- **GPU4** to **GPU5**
- **GPU6** to **GPU7**

**Sending to CPU:** Red arrows

**Receiving from CPU:** Blue arrows
The reduction:

- **ZHEMM**
- **ZHER2K**

- Receiving from CPU
- Sending to GPU locally
- Sending to CPU
- Receiving from GPU locally
- Communication
## MAGMA: first stage, reduction to band

The reduction: **ZHETRD_HE2HB** (time sec)

<table>
<thead>
<tr>
<th></th>
<th>1GPU</th>
<th>2GPU</th>
<th>3GPU</th>
<th>4GPU</th>
<th>5GPU</th>
<th>6GPU</th>
<th>7GPU</th>
<th>8GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>18</td>
<td>15</td>
<td>13.6</td>
<td>13.9</td>
<td>14.3</td>
<td>14.9</td>
<td>15.8</td>
<td>17.1</td>
</tr>
<tr>
<td>v2</td>
<td>17.1</td>
<td>9.4</td>
<td>8.2</td>
<td>8.4</td>
<td>8.1</td>
<td>8.1</td>
<td>8.5</td>
<td>8.7</td>
</tr>
<tr>
<td>v3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>61</td>
<td>43</td>
<td>36.7</td>
<td>34</td>
<td>33.2</td>
<td>33.5</td>
<td>34.4</td>
<td>36.8</td>
</tr>
<tr>
<td>v2</td>
<td>60</td>
<td>28.8</td>
<td>21.8</td>
<td>21.2</td>
<td>20.4</td>
<td>20.1</td>
<td>19.9</td>
<td>20.1</td>
</tr>
<tr>
<td>v3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The reduction:

CPU: QR on panel $i$

CPU: waiting next panel $i+1$

GPU: compute $W(i)$ and update next panel $(i+1)$

GPU: update trailing matrix

0: $NT =$ number of block
1: for $i = 0; 1$ to $NT-1$
2: panel$_{step_i}$: $QR(i) \Rightarrow (V,T)$
3: compute $X = AVT$ (SYMM)
4: compute $W = X - (\frac{1}{2})V^*T^*V^*X$
5: update trailing matrix $A = A - W^*V - V^*W$ (SYR2K)
6: end for
MAGMA: first stage, reduction to band

The reduction: ZHEMM way2:

\[ x = \text{allreduce} \]

needs sum

64
MAGMA: first stage, reduction to band

The reduction: \texttt{ZHEMM way3}: \( C(\text{myblk}) = A(:,\text{myblk})' \ast B(:) \)

need assemble

\[
\begin{bmatrix}
0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{X} \\
\end{bmatrix}
\]

= 

\[
\begin{bmatrix}
\text{X} \\
\end{bmatrix}
\]
MAGMA: first stage, reduction to band

The reduction: ZHEMM DSYMM
MAGMA: first stage, reduction to band

The reduction: ZHEMM DSYMM

GPU0  GPU1
GPU2  GPU3

GPU4  GPU5
GPU6  GPU7

sending to CPU
receiv from CPU
MAGMA: first stage, reduction to band

The reduction: ZHEMM DSYMM

The diagram shows a network of GPUs and a CPU involved in the MAGMA reduction process. The GPUs (GPU0, GPU1, GPU2, GPU3, GPU4, GPU5, GPU6, GPU7) are connected in a network, with some GPUs sending data to the CPU and others receiving data from the CPU. The ZHEMM and DSYMM operations are highlighted as the core of the reduction process.
MAGMA: first stage, reduction to band

The reduction: \texttt{comm}

- \texttt{ZHEMM}
- \texttt{ZHER2K}
MAGMA: first stage, reduction to band

The reduction: \texttt{ZHETRD\_HE2HB (sec)}

<table>
<thead>
<tr>
<th></th>
<th>1GPU</th>
<th>2GPU</th>
<th>3GPU</th>
<th>4GPU</th>
<th>5GPU</th>
<th>6GPU</th>
<th>7GPU</th>
<th>8GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>18</td>
<td>15</td>
<td>13.6</td>
<td>13.9</td>
<td>14.3</td>
<td>14.9</td>
<td>15.8</td>
<td>17.1</td>
</tr>
<tr>
<td>v2</td>
<td>17.1</td>
<td>9.4</td>
<td>8.2</td>
<td>8.4</td>
<td>8.1</td>
<td>8.1</td>
<td>8.5</td>
<td>8.7</td>
</tr>
<tr>
<td>v3</td>
<td>23</td>
<td>12</td>
<td>8.8</td>
<td>7.5</td>
<td>6.9</td>
<td>6.5</td>
<td>6.8</td>
<td>6.8</td>
</tr>
<tr>
<td>15K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v1</td>
<td>61</td>
<td>43</td>
<td>36.7</td>
<td>34</td>
<td>33.2</td>
<td>33.5</td>
<td>34.4</td>
<td>36.8</td>
</tr>
<tr>
<td>v2</td>
<td>60</td>
<td>28.8</td>
<td>21.8</td>
<td>21.2</td>
<td>20.4</td>
<td>20.1</td>
<td>19.9</td>
<td>20.1</td>
</tr>
<tr>
<td>v3</td>
<td>67</td>
<td>30</td>
<td>21</td>
<td>19</td>
<td>16.3</td>
<td>16.2</td>
<td>16.2</td>
<td>16.2</td>
</tr>
</tbody>
</table>
**MAGMA: toward fast Eigensolver**

- **Characteristics**
  - **Stage 1**: BLAS-3, increasing computational intensity,
  - **Stage 2**: BLAS-1.5, new cache friendly kernel,
  - **4X/12X faster** than standard approach,
  - **Bottleneck**: if all Eigenvectors are required, it has 1 back transformation extra cost.

---

- **flops formula**: $4n^3/3 \times \text{time}$
  - **Higher is faster**

---

Keeneland
**GPU** M2090 (@1.3 GHz, peak 583 GFlop/s)
**CPU** Intel Xeon X5660@2.80GHz (2 x 6 cores)
MAGMA: toward fast Eigensolver

Characteristics
• **Kepler 1.5X** faster for double precision,
• **Kepler twice faster** for double complex precision,

flops formula: $4n^3/3 \times \text{time}$
Higher is faster

Keeneland
GPU M2090 (@1.3 GHz, peak 583 GFlop/s)
CPU Intel Xeon X5660@2.80GHz (2x6 cores)

GPU K20c
(13 MP@0.7 GHz, peak 1165 GFlop/s)
CPU Genuine Intel
(2x8 @2.60GHz, peak 333 GFlop/s)
MAGMA: the Eigenproblem algorithms

Idea:

- Develop similar approach for hybrid architectures (CPU+GPU)
- The idea is to dump expensive operations into GPU and try to overlap with the CPU.
MAGMA: toward fast Eigensolver

Performance comparison 1GPU

Matrix \( n = 14000 \)

Experiments on socket of six-Intel Xeon X5650 + Fermi M2090.
MAGMA: toward fast Eigensolver 10%

Performance comparison multi-GPU

Matrix \( n = 14000 \)

CSCS system, using one node
8 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)
Future work

Road map and open questions:

• Develop a panel on GPU (ongoing integration)
• Develop similar approach for SVD (ongoing).
• Developing a multi-GPU/multicore/distributed version of the algorithm (ongoing).
• Hessenberg, (Piotr)
  • Bulge chasing
  • Gaussian reduction
  • Sign functions

Thank you for your attention