Tiled QR Decomposition and Its Optimization on CPU and GPU Computing System

Dongjin Kim and Kyu-Ho Park

Presentation by Dongjin Kim

Ph.D. Student, CORE lab., Electrical Engineering, KAIST
djkim@core.kaist.ac.kr
October 1st, 2013
@ P2S2-2013
Contents

1. Introduction
2. Background
3. Motivation
4. Design
5. Evaluation
6. Conclusion
Heterogeneous Core System

- Common to use **heterogeneous cores** for performance
  - As distributed-memory system
- Properties
  1. Performance heterogeneity
     - Different computation speed
  2. Explicit memory copy needed
  3. GPGPUs expect a larger input than CPUs
     - Much more parallel cores than CPU
Performance Decreasing Factors

- Different computation environments
  - Core architecture, clock speed, memory bandwidth, ...
  - Some jobs can be calculated faster on CPU
    - Jobs with low-parallelism

- Need of explicit memory copy
  - CPU and GPU cannot access each other’s memory directly
  - Too many data to share → communication bottleneck → low utilization
QR Decomposition

• QR Decomposition: $A = QR$
  • $Q$: Orthogonal matrix
  • $R$: Upper triangular matrix

• Tiled QR decomposition - for parallelization
  • Triangulation: Make upper triangle for a tile (T)
  • Elimination: Make zero matrix for T-ed tile from another T-ed tile (E)

• Update-T: Update for right columns after T ($uT$)
• Update-E: Update for right columns after E ($uE$)
2. Background

DAG of Tiled QR Decomposition

- Triangulation leads …
  - Elimination
  - Update for Triangulation

- Elimination leads …
  - Update for Elimination

- Update for Elimination leads …
  - Triangulation (next column)
Load Change within Each QR Step

- Calculation time
  - Two update processes are faster than Triangulation or Elimination

- Parallelism
  - Two update processes have much more tiles to be calculated

- \( \rightarrow \) Separate Updates and Triangulation/Elimination on separated devices

### Table: The number of tiles to be operated

<table>
<thead>
<tr>
<th>Step</th>
<th>Num. tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangulation</td>
<td>( M )</td>
</tr>
<tr>
<td>Elimination</td>
<td>( M )</td>
</tr>
<tr>
<td>Update for triangulation</td>
<td>( M \times (N - 1) )</td>
</tr>
<tr>
<td>Update for elimination</td>
<td>( M \times (N - 1) )</td>
</tr>
<tr>
<td>(Remaining part ( M ) by ( N ))</td>
<td></td>
</tr>
</tbody>
</table>

〈Single tile operation on GTX680〉
Heterogeneity of Computing Devices

- Heterogeneous environment
  - Different architecture, clock speed, …

- Triangulation and Elimination
  - Less tiles than Updates
  - More computing power for a tile
  - \( \rightarrow \) Device’s speed!

- Update processes
  - More tiles
  - Less computing power for a tile
  - \( \rightarrow \) Device’s parallelism!

\( \rightarrow \) Find appropriate device

\( \langle \text{The number of tiles to be operated} \rangle \)

\( \langle \text{Single tile operation on GTX680} \rangle \)
Effect of the Number of Devices

- More data transfer time if the number of devices increases
- Trade-off between more parallel threads vs. comm. overhead
- → Find optimal number of devices for given matrix
Contributions

- **Optimize** tile distribution and the tiled QR decomposition operation *mathematically*
  - Divided QR decomposition steps into appropriate computing devices
    - Depending on the processing properties
  - Optimize the number of devices that participate in the tiled QR decomposition
    - Depending on processing speed and communication cost
  - Tile distribution based on the parallelism of each device
Main Computing Device Selection

- Main Computing Device
  - Mainly executes the **triangulation** and **elimination** processes
- How to select
  - Can it finish its job before other’s update processes?
    - Pre-processing $\rightarrow$ measure each device’s calculation time
    - Multiply the number of tiles to be calculated
    - Determine whether a device can finish its job before others
    - From above, select a device that has less parallel cores
      - Since T/E have lower parallelism

![Diagram of Main Computing Device Selection](image)
The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

\[ T_{op}(p) = \max_{1 \leq i \leq p} \left[ \begin{array}{c} \#tile_m \times (time_m(T) + time_m(E)) + \#tile(m) \times (time_m(UT) + time_m(UE)) \\ \text{or} \\ \#tile(i) \times (time_i(UT) + time_i(UE)) \end{array} \right] \]
The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

\[
T_{op}(p) = \max_{1 \leq i \leq p} \left[ \#\text{tile}_m + \#\text{tile}(m) \times \frac{\text{time}_m(T) + \text{time}_m(E)}{\text{time}_m(UT) + \text{time}_m(UE)} \right] \\
\#\text{tile}(i) \times \frac{\text{time}_i(UT) + \text{time}_i(UE)}
\]

- The number of tiles, distributed to each device
- Time taken for each step on each device

4. Design

The number of tiles, distributed to each device

Time taken for each step on each device
The Number of Devices Selection (1)

• Find best number of devices
  • To optimize trade-off between communication and parallelism

• How to select
  • Sort devices in descending order of update process speed
    • With the main computing device at the first
  • For all available devices, calculate expected operation time

\[
T_{op}(p) = \max_{1 \leq i \leq p} \left[ \begin{array}{c}
#tile_m \times (time_m(T) + time_m(E)) \\
+ #tile(m) \times (time_m(UT) + time_m(UE)) \\
\end{array} \text{ or } \right.
\]

\[
#tile(i) \times (time_i(UT) + time_i(UE))
\]

Expected time for main computing device
The Number of Devices Selection (1)

- Find best number of devices
  - To optimize trade-off between communication and parallelism
- How to select
  - Sort devices in descending order of update process speed
    - With the main computing device at the first
  - For all available devices, calculate expected operation time

\[
T_{op}(p) = \max_{1 \leq i \leq p} \left[ \frac{\#tile_m \times (time_m(T) + time_m(E)) + \#tile(m) \times (time_m(UT) + time_m(UE))}{\#tile(i) \times (time_i(UT) + time_i(UE))} \right]
\]
The Number of Devices Selection (2)

• How to select (cont’d)
  • For all available devices, calculate expected communication time

\[
T_{\text{comm}}(p) = \sum_{i=1}^{p} \left( 3MT^2 \times \text{size(element)} \times \frac{1}{\text{speed}(m, p)} \right) \\
+ (M - 1)T^2 \times \text{size(element)} \times \frac{1}{\text{speed}(j, m)}.
\]
How to select (cont’d)

For all available devices, calculate expected communication time

\[ T_{\text{comm}}(p) = \sum_{i=1}^{n} \left( 3MT^2 \times \text{size(element)} \times \frac{1}{\text{speed}(m, p)} \right) + (M - 1)T^2 \times \text{size(element)} \times \frac{1}{\text{speed}(j, m)} \]
The Number of Devices Selection (2)

- How to select (cont’d)
  - For all available devices, calculate expected communication time

\[
T_{\text{comm}}(p) = \sum_{i=1}^{p} \left( 3MT^2 \times \text{size(element)} \times \frac{1}{\text{speed}(m, p)} \right) - (M - 1)T^2 \times \text{size(element)} \times \frac{1}{\text{speed}(j, m)}
\]

Expected time for Triangulation and Elimination

\(MT\): Result Q matrices of Triangulation
\(2MT\): Result Q matrices of Elimination
The Number of Devices Selection (2)

- How to select (cont’d)
  - For all available devices, calculate expected communication time

\[
T_{comm}(p) = \sum_{i=1}^{p} \left( 3MT^2 \times size(element) \times \frac{1}{speed(m, p)} \right) \\
+ (M - 1)T^2 \times size(element) \times \frac{1}{speed(j, m)}
\]

Expected time for next column tiles
How to select (cont’d)

For all available devices, calculate expected communication time

\[ T_{comm}(p) = \sum_{i=1}^{p} \left( 3MT^2 \times \text{size(element)} \times \frac{1}{\text{speed}(m,p)} \right) + (M - 1)T^2 \times \text{size(element)} \times \frac{1}{\text{speed}(j,m)}. \]

Find \( p \) which minimizes \( T_{op}(p) + T_{comm}(p) \), \( 1 \leq p \leq N \).
Tile Distribution

• Distribute tiles on each device
  • All devices should finish its job synchronously to maximize performance

• Load balancing based on distribution guide array
  • An array consists of device IDs
  • Find integer ratio of all devices, based on the number of tiles to be processes on fixed time
    • Device ID 0,1,2 and performance 3:2:1 \(\rightarrow [0,1,2,0,1,0]\)
    • The count of each ID is proportional to the performance
  • Distribute each column tile

\[
\text{distribute}(i) = \text{guide\_array}[i\%\text{length}(\text{guide\_array})].
\]
Implementation

- **Manager thread**
  - Select main computing device, decide the number of participating devices, distribute tiles, and migrate dependent data

- **Computing thread**
  - Do its own job
  - Have multiple slave threads for parallel operation
Evaluation Environment

• CPU
  • Intel i7-3820 (Quad core, 3.6GHz)
• Main Memory
  • 32GB
• GPU
  • Two GTX680 (1536 cores) + one GTX580 (512 cores)
• OS
  • Ubuntu 12.04, with Linux 3.2.0
• GPU driver version
  • 304.54
• CUDA version
  • 5.0
Scalability

- Time taken for...
  - Only CPU: 4 cores
  - CPU+1GPU: 516 cores
  - CPU+2GPUs: 2,052 cores
  - CPU+3GPUs: 3,588 cores

Total operation time proportionally decreases
5. Evaluation

Effect of Main Computing Device Selection

- Total operation time, with changing the main computing device selection
  - With our algorithm: GTX580 was selected as main computing device
  - 13% speed-up with another GPU as main computing device
  - 5% speed-up without specific main computing device
Effect of The Number of Devices Selection

- Compare predicted optimal number and actual optimal number
- Our algorithm can find actual optimal number of devices

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>320</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>480</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>640</td>
<td>1.02</td>
<td>1.21</td>
</tr>
<tr>
<td>800</td>
<td>1.03</td>
<td>1.54</td>
</tr>
<tr>
<td>960</td>
<td>1.03</td>
<td>1.80</td>
</tr>
<tr>
<td>1120</td>
<td>1.06</td>
<td>2.35</td>
</tr>
<tr>
<td>1280</td>
<td>1.09</td>
<td>2.71</td>
</tr>
<tr>
<td>1440</td>
<td>1.13</td>
<td>2.68</td>
</tr>
<tr>
<td>1600</td>
<td>1.15</td>
<td>2.71</td>
</tr>
<tr>
<td>1760</td>
<td>1.18</td>
<td>2.63</td>
</tr>
<tr>
<td>1920</td>
<td>1.20</td>
<td>3.00</td>
</tr>
<tr>
<td>2080</td>
<td>1.22</td>
<td>3.52</td>
</tr>
<tr>
<td>2240</td>
<td>1.23</td>
<td>3.27</td>
</tr>
<tr>
<td>2400</td>
<td>1.25</td>
<td>3.07</td>
</tr>
<tr>
<td>2560</td>
<td>1.26</td>
<td>3.02</td>
</tr>
<tr>
<td>2720</td>
<td>1.28</td>
<td>3.12</td>
</tr>
<tr>
<td>2880</td>
<td>1.31</td>
<td>3.07</td>
</tr>
<tr>
<td>3040</td>
<td>1.34</td>
<td>3.20</td>
</tr>
<tr>
<td>3200</td>
<td>1.36</td>
<td>3.18</td>
</tr>
<tr>
<td>3360</td>
<td>1.38</td>
<td>3.17</td>
</tr>
<tr>
<td>3520</td>
<td>1.40</td>
<td>2.90</td>
</tr>
<tr>
<td>3680</td>
<td>1.42</td>
<td>2.82</td>
</tr>
<tr>
<td>3840</td>
<td>1.44</td>
<td>2.78</td>
</tr>
<tr>
<td>4000</td>
<td>1.46</td>
<td>2.69</td>
</tr>
</tbody>
</table>

(*) Normalized value for smallest time
5. Evaluation

Effect of Tile Distribution

- Check the performance with Distribution Guide Array
- 21% faster than evenly distributed case
- 10% faster than distribution just based on the number of cores
6. Conclusion

Conclusion

• Summary
  • Mathematical optimization for tile QR decomposition
    • On CPU and GPU heterogeneous computing system
  • Select a specific device as the main computing device
    • Handles Triangulation and Elimination
  • The number of device optimization
  • Distribution based on distribution guide array
  • Algorithms can optimize the performance

• Further works
  • Considering very large matrix operation
    • Lack of memory problem will appear
  • Expand algorithms into other computing systems
    • Generalization
Thank YOU!

Any Question?