Optimal Block Size for Matrix Multiplication Using Blocking

Sasko Ristov, Marjan Gushev – Ss. Cyril and Methodius University, FSCE
{marjan.gushev, sashko.ristov}@finki.ukim.mk

Goran Velkoski – Innovation LLC
goran.velkoski@innovation.com.mk
Abstract

• **Speeding up** the multiplication of huge matrices is imperative

• **Blocking reduces the cache misses**
  – choosing the block size is not the only optimization

• This paper analyzes the impact of various block size ($M \times K$ and $K \times N$) on the performance.
  – **Different** parameter values for $K$
  – **predefined** values of the parameters $M$ and $N$,
  – test the algorithm behavior in different cache regions.
Abstract

• The results of the experiments show three phenomena.
  – if $M > N$, then choosing the block $M \times N$ of the first matrix will achieve a significantly greater speed.
  – if the second parameter $N$ is increased for constant $M$ has no significant influence on the performance.
  – the speed decreases significantly if $N$ is increasing for constant $M$. 
Outline

• Background & Motivation
• Testing Methodology
• The Results of the Experiments
• Discussion
• Conclusion & Future Work
Background – Cache Associativity vs. Blocking

• Blocking algorithm improves the MM
  – Only L1 cache size
  – Does not takes into account the cache set associativity problem !!!


Background – Improved Blocking

• Recently we proposed 1D/2D blocking MM
  – Better for AMD (low associativity)


• In this paper, determine the optimal block dimensions $M \times K$ and $K \times N$
  – the same number of operations is executed
  – Improve memory access time
Outline

- Background & Motivation
- Testing Methodology
- The Results of the Experiments
- Discussion
- Conclusion & Future Work
Testing Algorithms

- Matrix elements are **double**
- Chose values to **mitigate associativity problem**
  - $56 = 64 - 8$
  - $224 = 56 \times 4$
  - $896 = 56 \times 16$

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Notation</th>
<th>$BA_{M \cdot K}$</th>
<th>$BB_{K \cdot N}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56x56</td>
<td>$56 \cdot K$</td>
<td>$K \cdot 56$</td>
</tr>
<tr>
<td>2</td>
<td>56x224</td>
<td>$56 \cdot K$</td>
<td>$K \cdot 224$</td>
</tr>
<tr>
<td>3</td>
<td>56x896</td>
<td>$56 \cdot K$</td>
<td>$K \cdot 896$</td>
</tr>
<tr>
<td>4</td>
<td>224x56</td>
<td>$224 \cdot K$</td>
<td>$K \cdot 56$</td>
</tr>
<tr>
<td>5</td>
<td>224x224</td>
<td>$224 \cdot K$</td>
<td>$K \cdot 224$</td>
</tr>
<tr>
<td>6</td>
<td>224x896</td>
<td>$224 \cdot K$</td>
<td>$K \cdot 896$</td>
</tr>
<tr>
<td>7</td>
<td>896x56</td>
<td>$896 \cdot K$</td>
<td>$K \cdot 56$</td>
</tr>
<tr>
<td>8</td>
<td>896x224</td>
<td>$896 \cdot K$</td>
<td>$K \cdot 224$</td>
</tr>
<tr>
<td>9</td>
<td>896x896</td>
<td>$896 \cdot K$</td>
<td>$K \cdot 896$</td>
</tr>
</tbody>
</table>

- Cover all cache regions
Testing Environment

- Intel Xeon CPU X5647 @ 2.93GHz with 8GB RAM memory.
  - quad core,
  - each core has its own private L1 (32KB) and L2 (256KB) caches.
  - All 4 cores share L3 cache of 12MB.
Test Data

- The execution time $T(M, N, K)$ changing $K$
- Calculate *Speed* $V(M, N, K)$

$$V(M, N, K) = \frac{2 \cdot M \cdot N \cdot K}{T(M, N, K)}$$
Experiments

- $M < N$

- $M > N$
Three Test Goals

• The First Goal
  – determine which pair of block sizes provides better performance if the blocks have the same number of elements, but exchanged parameters.
  – For example
    • \( M = 56 < N = 224 \)
    • \( M = 6 < N = 9 \).
  – The parameter \( K \) is variable.
Three Test Goals

• The **second** goal
  – determine the **impact of** the parameter $M$

• The **third** goal
  – determine the **impact of** the parameter $N$

• **Varying** the parameter $K$
Outline

• Background & Motivation
• Testing Methodology
• The Results of the Experiments
• Discussion
• Conclusion & Future Work
Horizontal or Vertical Rectangle?!

- Similar results for all cases
- Both curves are identical until some $K$
  - the point when both matrices together exceed L2 cache (Private per core)
- Better when $M > N$
The $M$'s Impact

- Increasing $M$ does not impact the performance when matrices can be placed in cache.
- Similar results for all three values of $N$. 
The $N$'s Impact

- **Increasing $N$ negatively impacts** the algorithm performance,
  - emphasized for greater $K$. 
Outline

• Background & Motivation
• Testing Methodology
• The Results of the Experiments
• Discussion
• Conclusion & Future Work
**Discussion**

- **Choosing** a rectangle **depends** only **when** matrices cannot be placed in L2 cache,
  - favoring the rectangle with **greater** $M$ - vertical rectangle.

- **VERY IMPORTANT:**
  - Speed is increased until L2 (Not until L1) **for horizontal** rectangle, while the speed **keeps its value for vertical** rectangle.

- **This leads to conclusion:**
  - the **blocking can be choosed** with **greater blocks** (Not in L1) since **the number of operations will be smaller**, and thus **the overall execution will be faster** than traditional blocking.
Discussion

• The impact of M and N is totally different
  – $M$ has a small impact to the performance, especially for smaller values of the parameter $N$.
    • Means that blocks can be even $> L1$ cache size.
  – Increasing the $N$ significantly reduces the algorithm performance

• The common
  – increased impact for greater $K$
Outline

• Background & Motivation
• Testing Methodology
• The Results of the Experiments
• Discussion
• Conclusion & Future Work
Conclusion & Future Work

- Three phenomena
- Discussed results
  - Greater than L1

- Other values of parameters $M$ and $N$
- Parallelization of these experiments using
  - multi-core CPUs,
  - GPUs (also have set associative caches)
THANK YOU FOR YOUR ATTENTION

• QUESTIONS?