The Performance Impact Analysis of Loop Unrolling

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

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

"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER
SCIENCE AND ENGINEERING

Abstract

- **Loop unrolling** usually speed ups a program with loops
  - **reduces the jump operations** at the end of the loops.
- This paper analyzes the **impact of loop unrolling** on **various processor** types and memory patterns.
- The results show **a huge correlation** of the **cache** and **problem size**.
- A **higher speedup** for a **smaller problem** size
  - does not impact a problem whose size is greater than the capacity of the last level cache size, due to the huge number of cache misses.
- The loop unrolling achieves **much greater speedup on Intel**, rather than AMD CPUs.

Outline

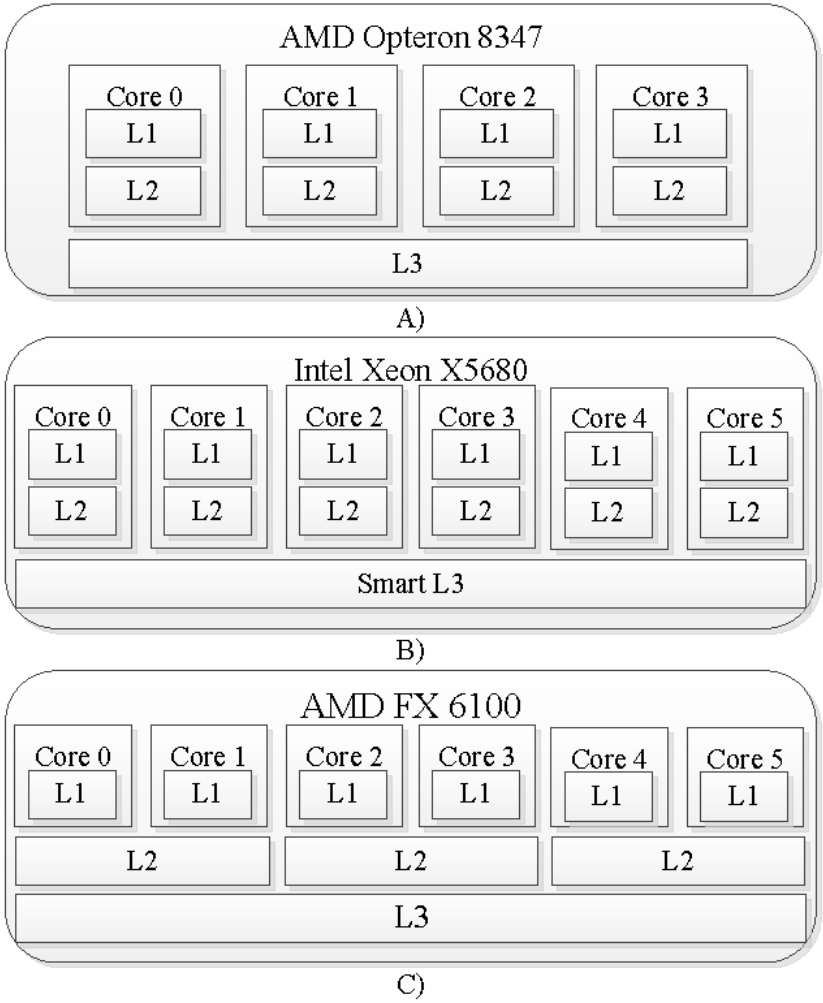
- **Testing Methodology**
- The Results of the Experiments
- Discussion
- Conclusion & Future Work

Testing Algorithms

- 5 Dense Matrix Matrix Multiplication implementations
  - **without loop** unrolling;
  - **2 loops** are unrolled;
  - **4 loops** are unrolled;
  - **8 loops** are unrolled; and
  - **16 loops** are unrolled.

Testing Environment

- Three architectures:
  - AMD **Opteron** 8347
    - Barcelona
  - Intel **Xeon** X5680
  - AMD FX 6100
    - Zambezi



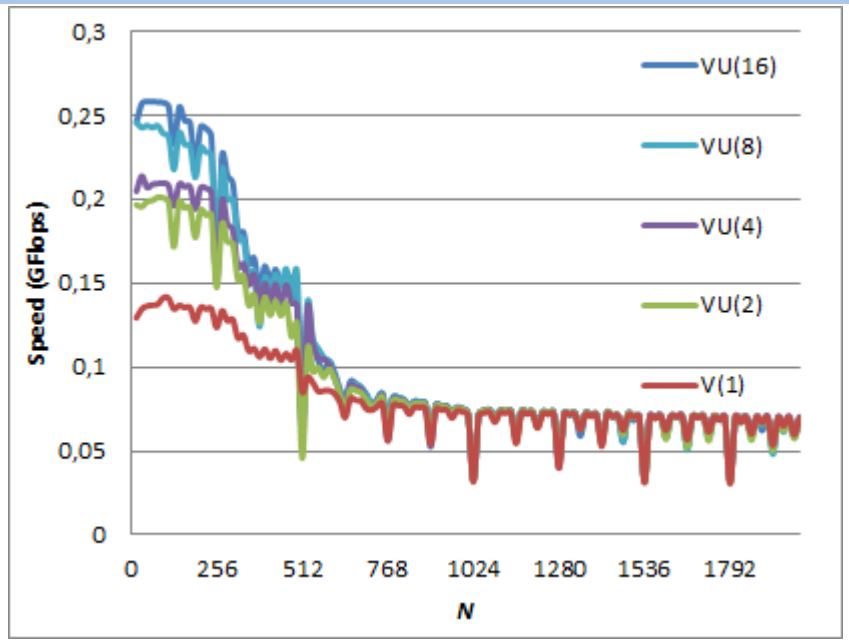
Test Data

- **Execution Time *T(U)*** is measured.
- ***Speed VU(U)*** is calculated
  - *U* denotes the bunch size of the unrolling
  - {1, 2, 4, 8, 16}
- ***Relative Speedup R(U)***
  - **compare the impact of unrolling bunch size** on a particular environment

Outline

- Testing Methodology
- **The Results of the Experiments**
- Discussion
- Conclusion & Future Work

AMD Opteron 8347

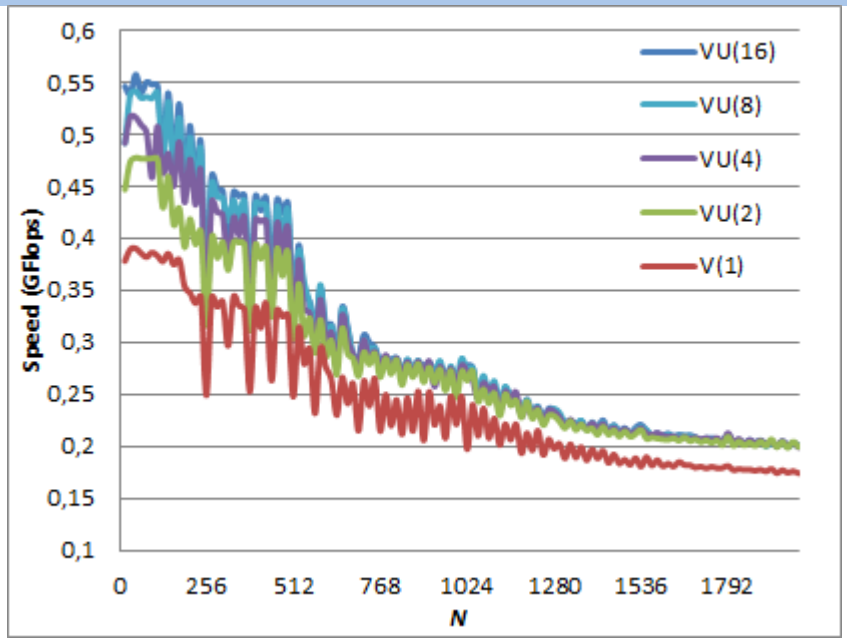


- **LEFT:**
  - **greater speed**
  - **speed difference**
- **RIGHT**
  - **no visible difference**
  - drawbacks at N = 512, 640, 768, … cache associativity problem (small associativity)

M. Gusev and S. Ristov, "**Performance Gains and Drawbacks using Set Associative Cache**", JNIT, Journal of Next generation Information Technology, ISSN: 2233-9388, 2012, Volume 3, Number 3, pp.87-98.

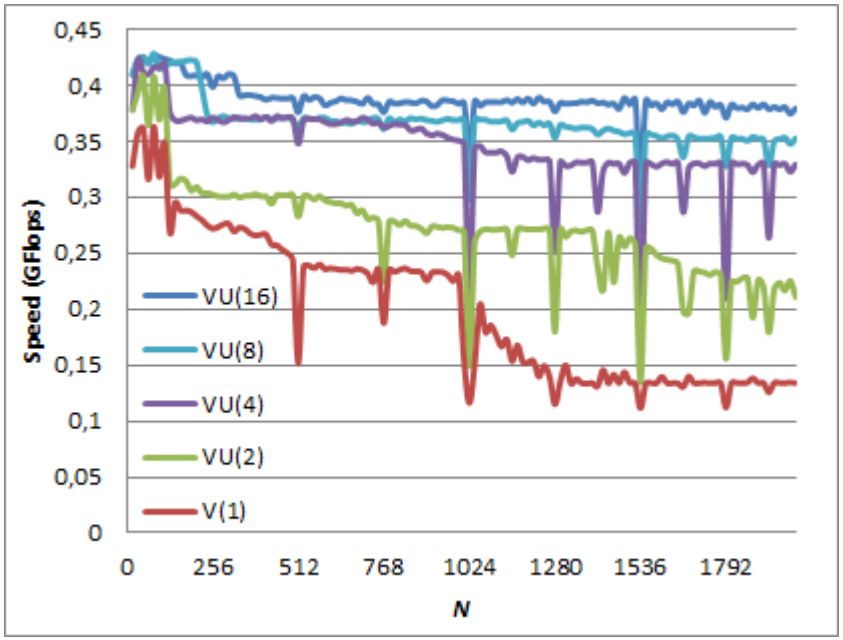
Intel Xeon X5680 CPU

- Improved speed with LU
- Similar to AMD
  - But the speed is also different in both regions



AMD FX 6100 CPU

- The loop unrolling is essential !!!
  - During the whole region N
  - stable speed for each test case for U=8 and U=16.

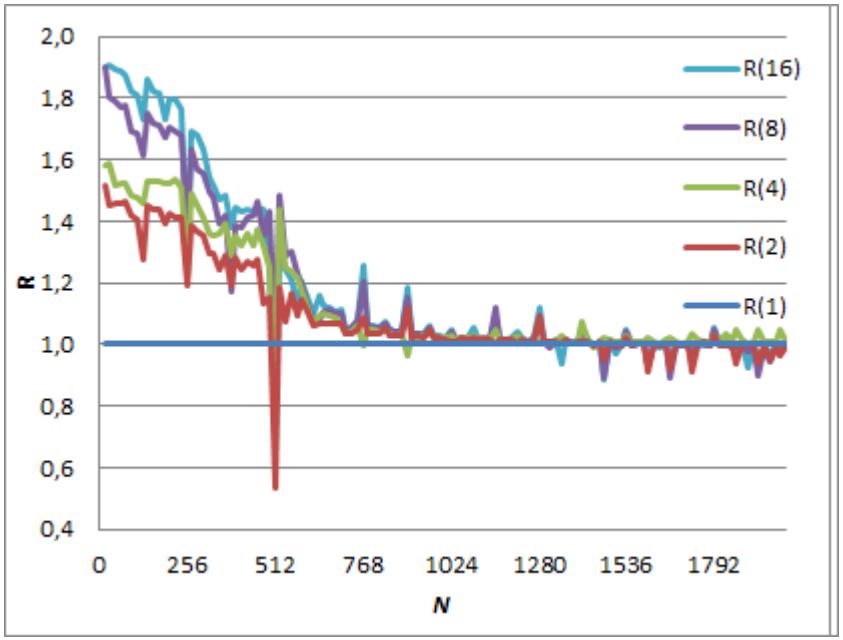


Outline

- Testing Methodology
- The Results of the Experiments
- **Discussion**
- Conclusion & Future Work

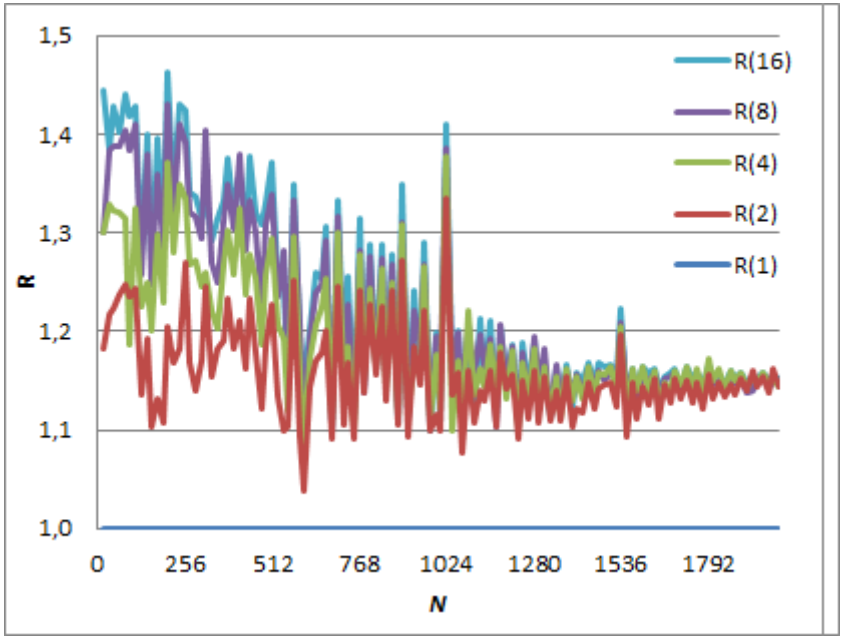
Bunch size impact on AMD Opteron 8347

- The **LU does not make** any **difference** when **matrices cannot be stored in cache**.
  - the response **mostly depends on the memory access**, rather than the operations.
- The speedup proportionally increases with the number of unrolled loops.



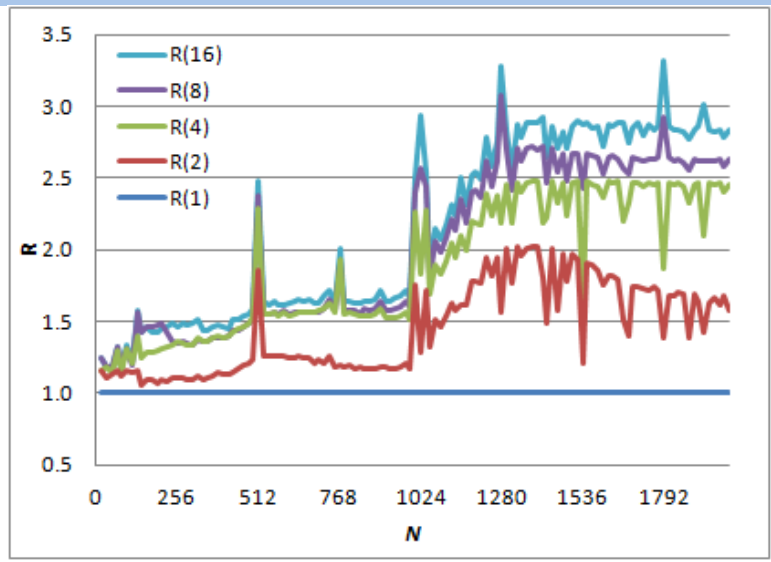
Bunch size impact on Intel Xeon X5680 CPU

- The speedup is **proportional to the number of unrolled loops** for smaller matrices
- it diverges to the value speedup when using bunch size 2 for greater matrices that cannot be placed in the last level L3 cache



Bunch size impact on AMD(tm) FX 6100 CPU

- Totally **opposite** results **on the Zambezi** architecture
- The **greater relative speedup** is recorded **for larger matrices** than for the smaller matrices.
- The larger number of unrolled loops results with increased speedup.
- LU optimization method provides better performance when the number of memory accesses and cache misses increase.



Outline

- Testing Methodology
- The Results of the Experiments
- Discussion
- **Conclusion & Future Work**

Conclusion

- We compared the **loop unrolling technique** for algorithm optimization using the dense matrix multiplication algorithm. We measured **the performance for three processors**.
- The experiments were conducted
  - varying the bunch size of loop unrolling
  - the matrix sizes.

Conclusion

- the LU speed ups the cache intensive algorithms for the greater bunch size
  - depends on the CPU architecture.
  - This is **more emphasized for smaller matrices** (can be placed in private per core cache memory) on AMD Opteron and Intel Xeon,
  - it is significantly emphasized **for greater matrices** on the **AMD Zambezi** architecture.

Future Work

- **mathematically modeling** the **dependency** of
  - **loop unrolling** optimization technique and
  - **its correlation with the cache memory** architectures and capacity.
- Generalize this theory with **other** most used **algorithms** that have loops
  - **dot product** with one loop, or
  - **matrix-vector** multiplication with two loops.

THANK YOU FOR YOUR ATTENTION

- QUESTIONS?