Automatic Cache Aware Roofline Model Building and Validation Using Topology Detection

Nicolas Denoyelle - Aleksandar Ilic - Brice Goglin - Leonel Sousa - Emmanuel Jeannot

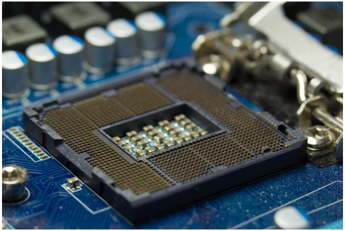
Inria (France) - INESC-ID (Portugal) - Inria (France) - INESC-ID (Portugal) - Inria (France)

October 4, 2016

1. Introduction
2. The Cache Aware Roofline Model
3. Tool and Validation
4. Conclusion & Future Work

Hardware Complexity is Growing

- ↗ Cores: up to 72 on Intel KNL.
- ↘ Memory per Core.
- ↗ Cache Sharing .
- ↗ Cache Hierarchy: several levels, side cache *etc.*. . .



1. Introduction
2. **The Cache Aware Roofline Model**
3. Tool and Validation
4. Conclusion & Future Work

The Cache Aware Roofline Model (CARM)

The Cache Aware Roofline Model (CARM) simplifies this for you !

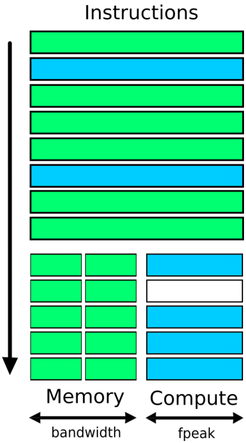
- What is the best you can get from the chip ? [Platform Model]
- Is it worth investing in code optimization ? [Application Model]

Platform Model

Improvement over the Original Roofline Model (ORM).

Assumptions:

- The machine is bound whether by the memory unit or by the compute unit.
- Both unit are not dependent on each other.

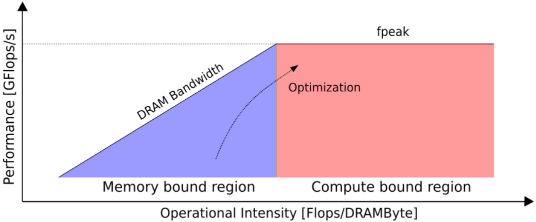


Application Model

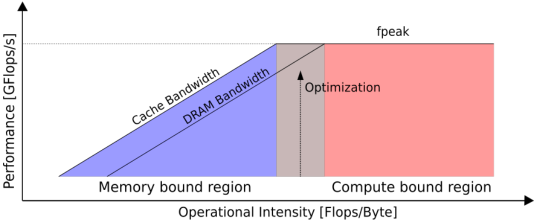
$$ddot = ddot + b[i] * c[i] \tag{1}$$

- memory: $b[i], c[i] = 8 + 8Bytes$
- compute: $+, * = 2flops$
- arithmetic intensity: $2^{-3}Flops/Bytes$
- performance: $Flops/Time$

ORM versus CARM



Bytes transferred from DRAM to last level cache



Bytes issued from Cores.

1 Introduction

2 The Cache Aware Roofline Model

3 **Tool and Validation**

4 Conclusion & Future Work

A Tool to Build and Validate the Model

Build Model from Portable Blocks

- Hwloc gives the machine informations: cache structure and attributes, cores.
- The compiler gives architecture type and let us select the best instruction set.

Micro-Benchmark Each Unit

- Bandwidth benchmark each cache level, and several stream types: Load, Store, 2Loads/Store.
- Floating Point peak benchmark: MUL, ADD, MAD, FMA.
- Micro Benchmarks of several arithmetic intensities for validation.

A Tool for Non Experts

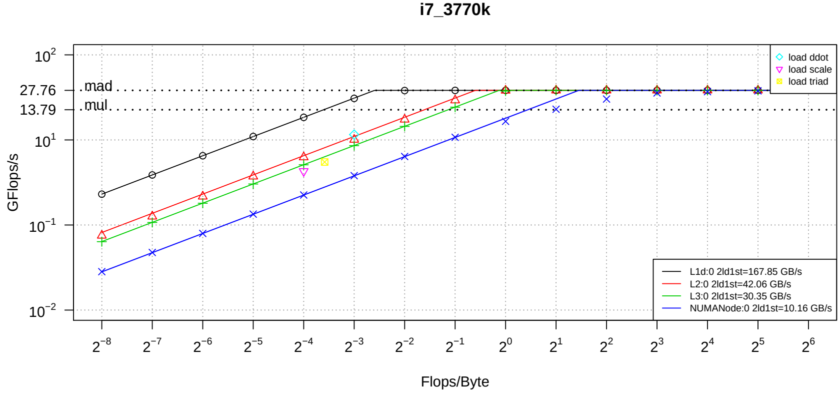
Easy steps

- Build the model for the target platform.
- Collect CARM metrics from your application with our library (Hardware counters: PAPI).
- Plot this on a chart and get insights on your application.

Quick Glance if the Model is Wrong

- Validation matches model ?
- Micro-Benchmarks matches theoretical throughput ?

Tool Output



i7_3770k
GFlops/s
Flops/Byte
10^2
27.76
13.79
10^1
10^{-1}
10^{-2}
mad
mul
2^{-8}
2^{-7}
2^{-6}
2^{-5}
2^{-4}
2^{-3}
2^{-2}
2^{-1}
2^0
2^1
2^2
2^3
2^4
2^5
2^6
◇ load ddot
▽ load scale
☒ load triad
L1d:0 2ld1st=167.85 GB/s
L2:0 2ld1st=42.06 GB/s
L3:0 2ld1st=30.35 GB/s
NUMANode:0 2ld1st=10.16 GB/s

1 Introduction

2 The Cache Aware Roofline Model

3 Tool and Validation

4 **Conclusion & Future Work**

Conclusion

We provide a Tool to build the CARM and model applications.
It is portable and easy to use. It reaches near theoretical architecture throughput.

Future Work

The CARM gives insight over Caches (i.e temporal locality).
We will use it to model system heteogeneous memory, and give insights on data locality.

Thank you !

https://github.com/NicolasDenoyelle/LARM-Locality-Aware-Roofline-Model-

Automatic Cache Aware Roofline Model Building and Validation Using Topology Detection

Nicolas Denoyelle - Aleksandar Ilic - Brice Goglin - Leonel Sousa - Emmanuel Jeannot

Inria (France) - INESC-ID (Portugal) - Inria (France) - INESC-ID (Portugal) - Inria (France)

October 4, 2016