Energy to Solution vs Time to Solution, towards energy-aware HPC applications

E. Calore[1] S. F. Schifano[1] R. Tripiccione[1]

[1] INFN Ferrara and Università degli Studi di Ferrara, Italy

COLA Workshop

Ferrara - February 25th, 2016

Outline

Outline

Introduction

- Energy efficiency is quickly gaining importance in the HPC field
- Despite of this, optimization efforts are still mainly committed to minimize the time-to-solution
- On the other side, in other fields such us embedded devices, optimization efforts are more strongly committed towards minimizing the energy-to-solution

In this work...

...we explore, for several High-End and Low-Power architectures, the offered opportunities to tune the energy consumption, highlighting possible tradeoffs between two metrics: time- and energy-to-solution.

Introduction

- Energy efficiency is quickly gaining importance in the HPC field
- Despite of this, optimization efforts are still mainly committed to minimize the time-to-solution
- On the other side, in other fields such us embedded devices, optimization efforts are more strongly committed towards minimizing the energy-to-solution

In this work...

...we explore, for several High-End and Low-Power architectures, the offered opportunities to tune the energy consumption, highlighting possible tradeoffs between two metrics: time- and energy-to-solution.

Introduction

- Energy efficiency is quickly gaining importance in the HPC field
- Despite of this, optimization efforts are still mainly committed to minimize the time-to-solution
- On the other side, in other fields such us embedded devices, optimization efforts are more strongly committed towards minimizing the energy-to-solution

In this work...

...we explore, for several High-End and Low-Power architectures, the offered opportunities to tune the energy consumption, highlighting possible tradeoffs between two metrics: time- and energy-to-solution.

Introduction

- Energy efficiency is quickly gaining importance in the HPC field
- Despite of this, optimization efforts are still mainly committed to minimize the time-to-solution
- On the other side, in other fields such us embedded devices, optimization efforts are more strongly committed towards minimizing the energy-to-solution

In this work...

...we explore, for several High-End and Low-Power architectures, the offered opportunities to tune the energy consumption, highlighting possible tradeoffs between two metrics: time- and energy-to-solution.

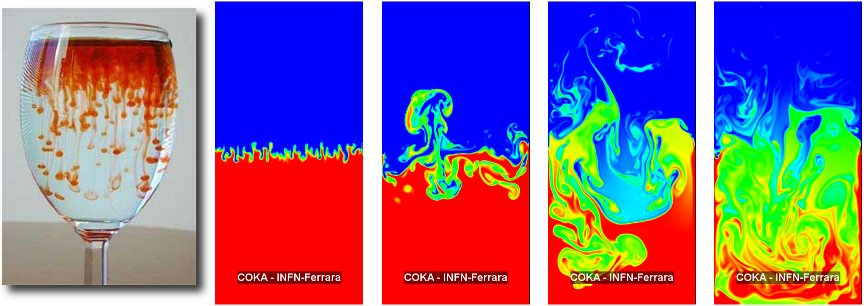
Outline

The D2Q37 Lattice Boltzmann Model

- Lattice Boltzmann method (LBM) is a class of computational fluid dynamics (CFD) methods
- LBM methods simulate a discrete **Boltzmann** equation, which under certain conditions, reduce to the **Navier-Stokes** equation
- **virtual particles** called **populations** arranged at edges of a discrete and regular grid are used to simulate a synthetic and simplified dynamics
- the interaction is implemented by two main functions applied to the virtual particles: **propagation** and **collision**
- D2Q37 is a D2 model with 37 components of velocity (populations)
- suitable to study behaviour of **compressible** gas and fluids optionally in presence of **combustion** effects
- correct treatment of Navier-Stokes, heat transport and perfect-gas ($P = \rho T$) equations

Simulation of the Rayleigh-Taylor (RT) Instability

Instability at the interface of two fluids of different densities triggered by gravity.



A cold-dense fluid over a less dense and warmer fluid triggers an instability that mixes the two fluid-regions (till equilibrium is reached).

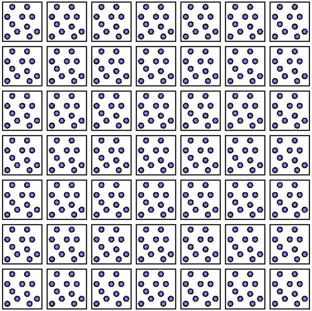
Computational Scheme of LBM

```
foreach time-step

  foreach lattice-point
    propagate();
  endfor

  foreach lattice-point
    collide();
  endfor

endfor
```



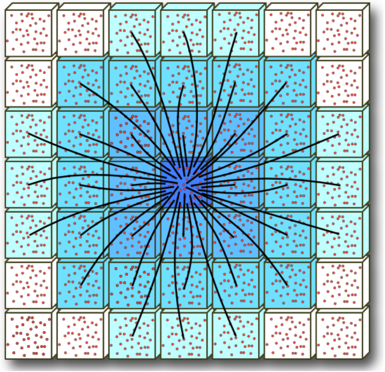
Embarassing parallelism

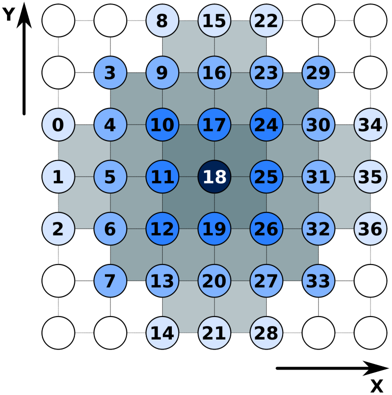
All sites can be processed in parallel applying in sequence propagate and collide.

Challenge

Design an efficient implementation able exploit a large fraction of available peak performance.

D2Q37: propagation scheme





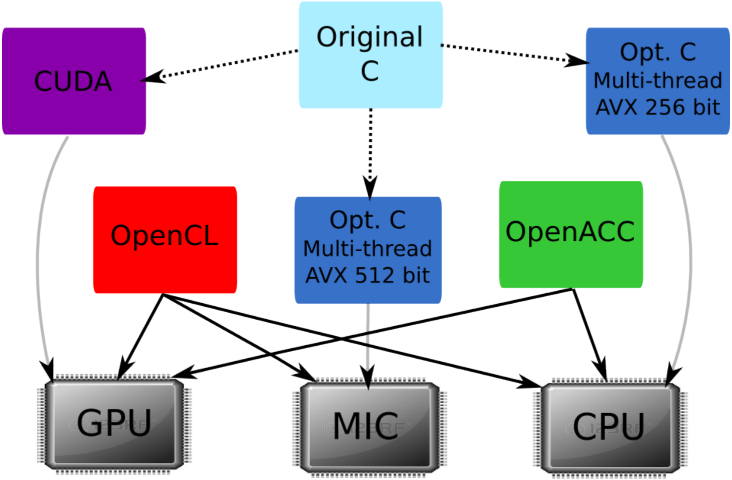
- perform accesses to neighbour-cells at distance 1,2, and 3
- generate memory-accesses with **sparse** addressing patterns

D2Q37 collision

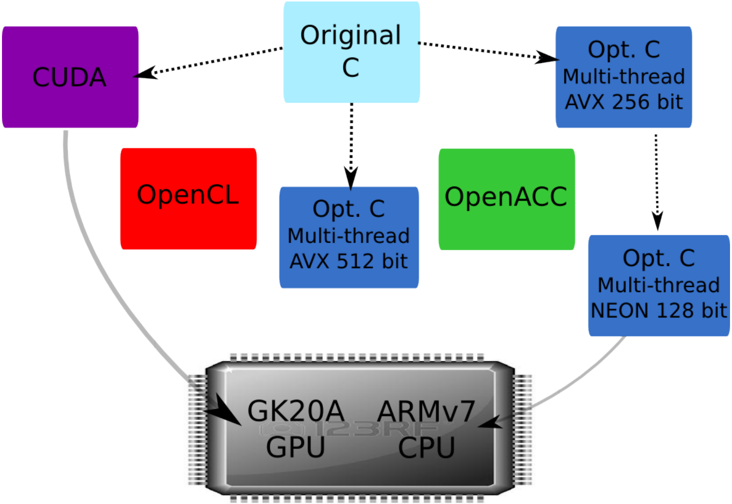
- collision is computed at each lattice-cell after computation of boundary conditions
- computational intensive: for the D2Q37 model requires $\approx$ 7500 DP floating-point operations
- completely local: arithmetic operations require only the populations associate to the site

- computation of propagate and collide kernels are kept separate
- after propagate but before collide we may need to perform collective operations (e.g. divergence of of the velocity field) if we include computations conbustion effects.

Initial Code implementations



Code implementations

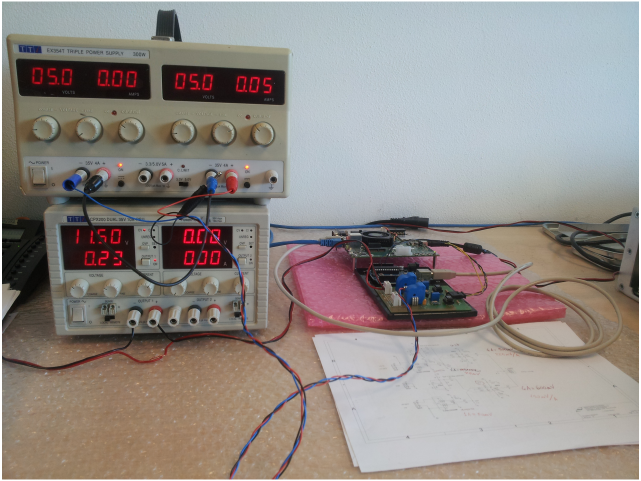


Outline

Outline

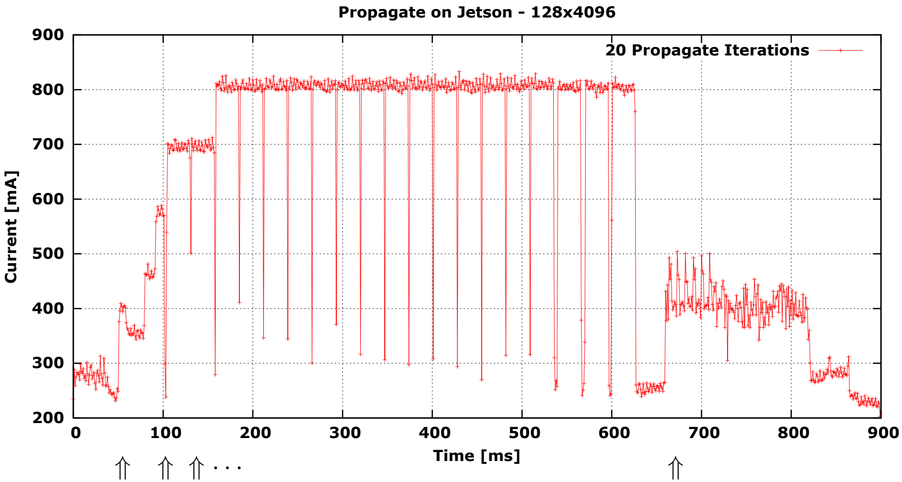
Setup to sample instantaneous current absorption

One current to voltage converter...
...plus an Arduino UNO (microcontroller + 10-bit ADC + Serial over USB)



Current to Voltage + Digitization with Arduino + USB Serial

Acquired data example with default frequency scaling

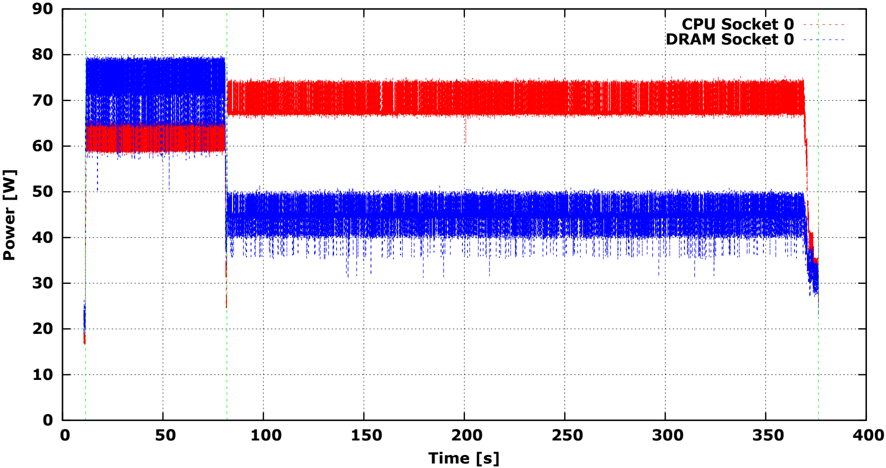


Iterations can be counted

This is a D2H transfer

Outline

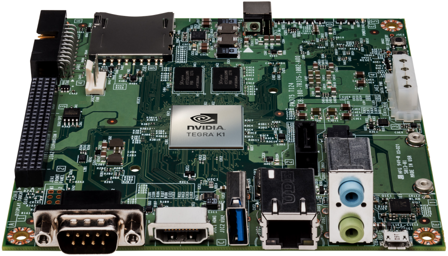
Acquired data example using RAPL counters



Intel Haswell CPU energy counters acquired at 100Hz and converted in Watt; acquisition performed with a custom developed wrapper to the PAPI library.

Outline

NVIDIA Jetson TK1



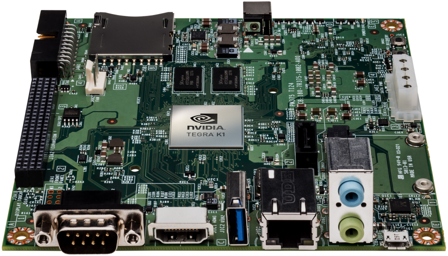
SoC: Tegra K1

- CPU: NVIDIA "4-Plus-1" 2.32GHz ARM quad-core Cortex-A15, with battery-saving shadow-core
- GPU: NVIDIA Kepler "GK20a" GPU with 192 SM3.2 CUDA cores

Awarded for the Best Paper

7^{th} Workshop on UnConventional High Performance Computing (UCHPC), Porto 2014

NVIDIA Jetson TK1



SoC: Tegra K1

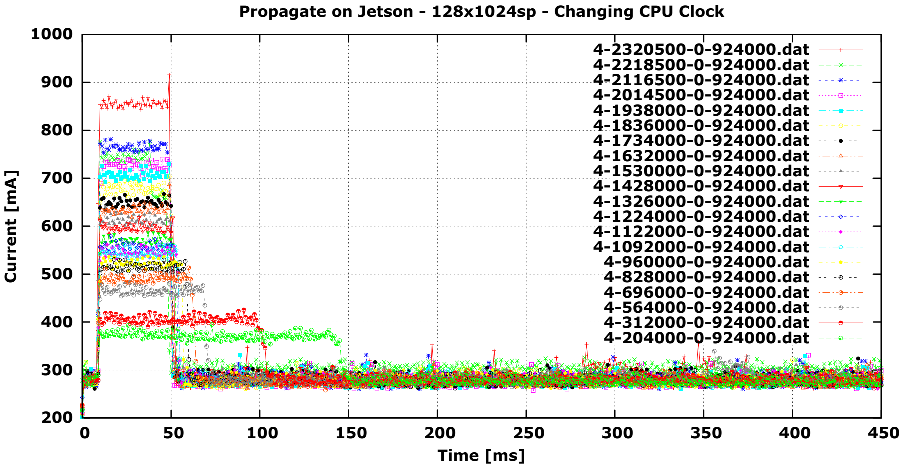
- CPU: NVIDIA "4-Plus-1" 2.32GHz ARM quad-core Cortex-A15, with battery-saving shadow-core
- GPU: NVIDIA Kepler "GK20a" GPU with 192 SM3.2 CUDA cores

Awarded for the Best Paper

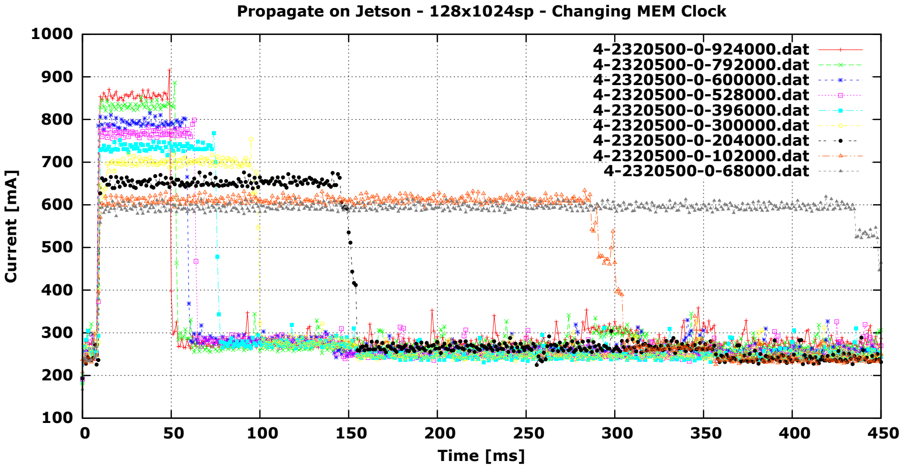
7^{th} Workshop on UnConventional High Performance Computing (UCHPC), Porto 2014

Outline

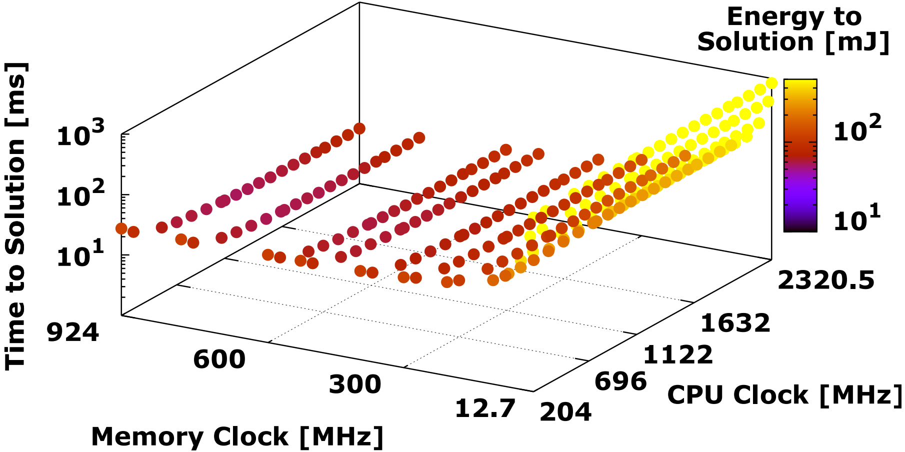
Propagate changing the G cluster clock



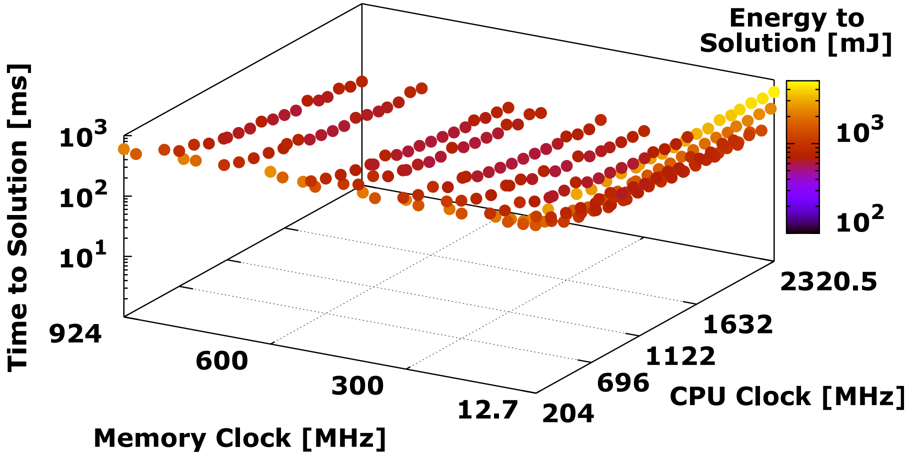
Propagate changing the MEM clock



Time and Energy to solution (Propagate)

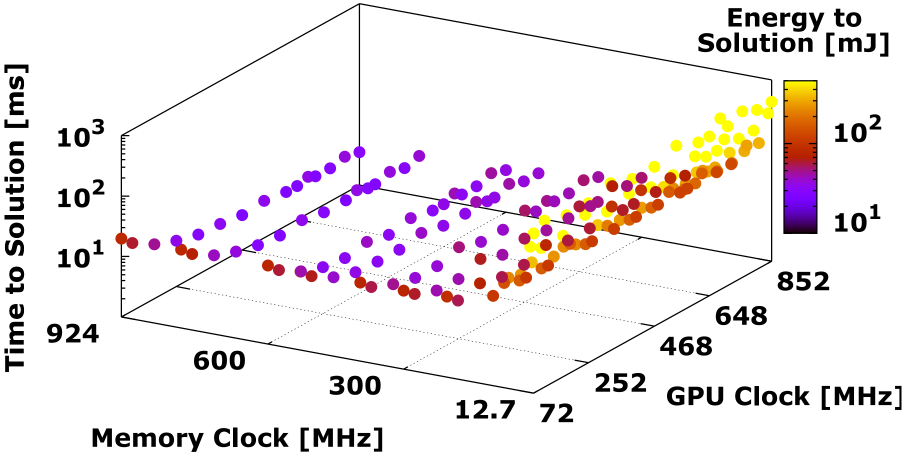


Time and Energy to solution (Collide)

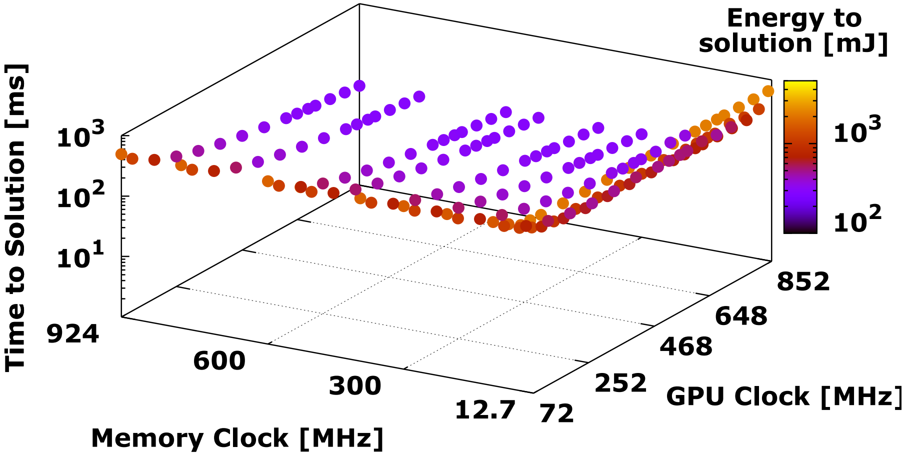


Outline

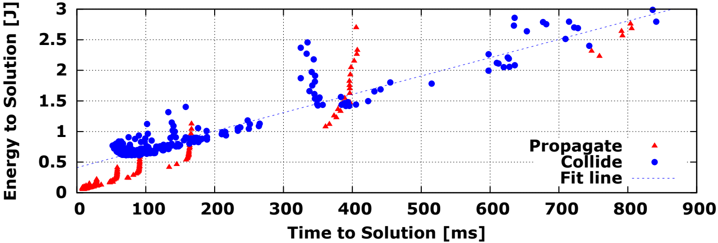
Time and Energy to solution (Propagate)

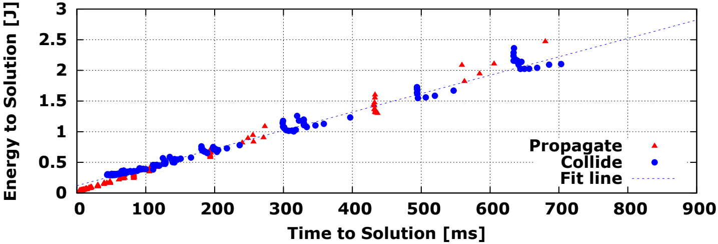


Time and Energy to solution (Collide)



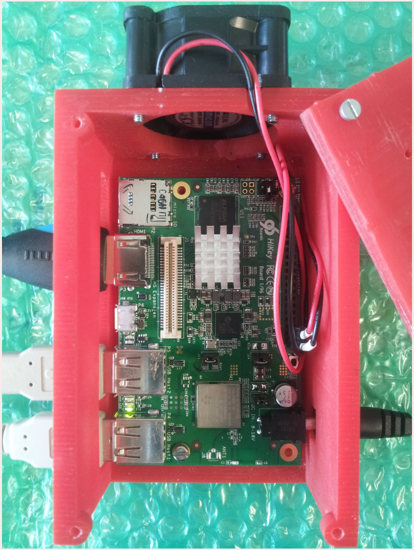
Energy to Sol. vs Time to Sol. CPU(top), GPU(bottom)





Outline

96Boards - HiKey



SoC: HiSilicon Kirin 6220

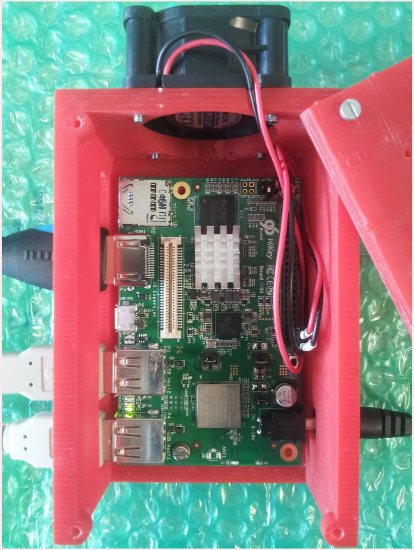
- CPU: 8 core ARM Cortex-A53 running at 1.2GHz (64-bit aarch64)
- GPU: ARM Mali 450-MP4 GPU
- MEM: 1GB of 800MHz LPDDR3

Awarded for the Best Paper

8th Workshop on UnConventional High Performance Computing (UCHPC), Vienna 2015

3D printed case to fit a fan (Thanks to V. Carassiti and A. Cotta Ramusino, INFN Ferrara)

96Boards - HiKey



SoC: HiSilicon Kirin 6220

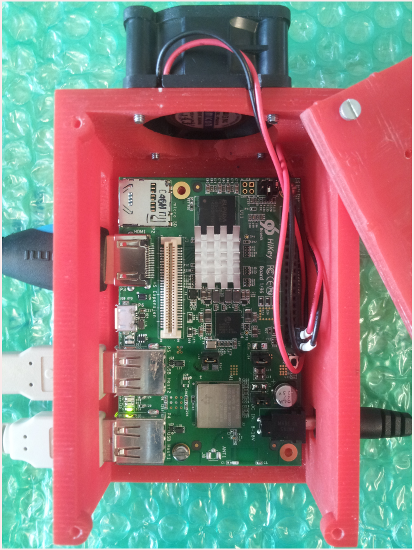
- CPU: 8 core ARM Cortex-A53 running at 1.2GHz (64-bit aarch64)
- GPU: ARM Mali 450-MP4 GPU
- MEM: 1GB of 800MHz LPDDR3

Awarded for the Best Paper

8th Workshop on UnConventional High Performance Computing (UCHPC), Vienna 2015

3D printed case to fit a fan (Thanks to V. Carassiti and A. Cotta Ramusino, INFN Ferrara)

96Boards - HiKey



SoC: HiSilicon Kirin 6220

- CPU: 8 core ARM Cortex-A53 running at 1.2GHz (64-bit aarch64)
- GPU: ARM Mali 450-MP4 GPU
- MEM: 1GB of 800MHz LPDDR3

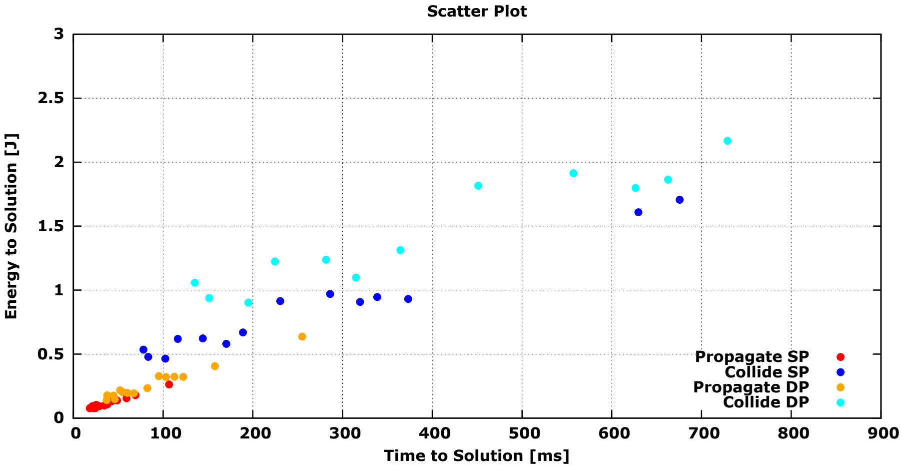
Awarded for the Best Paper

8th Workshop on UnConventional High Performance Computing (UCHPC), Vienna 2015

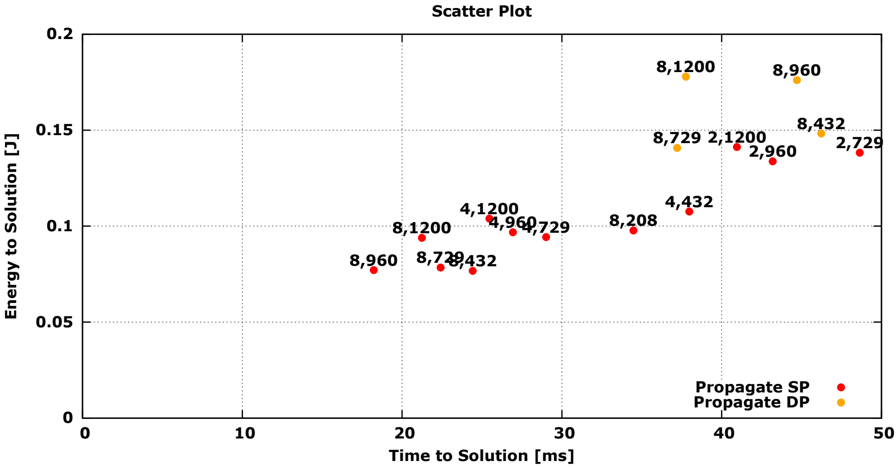
3D printed case to fit a fan
(Thanks to V. Carassiti and A. Cotta Ramusino, INFN Ferrara)

Outline

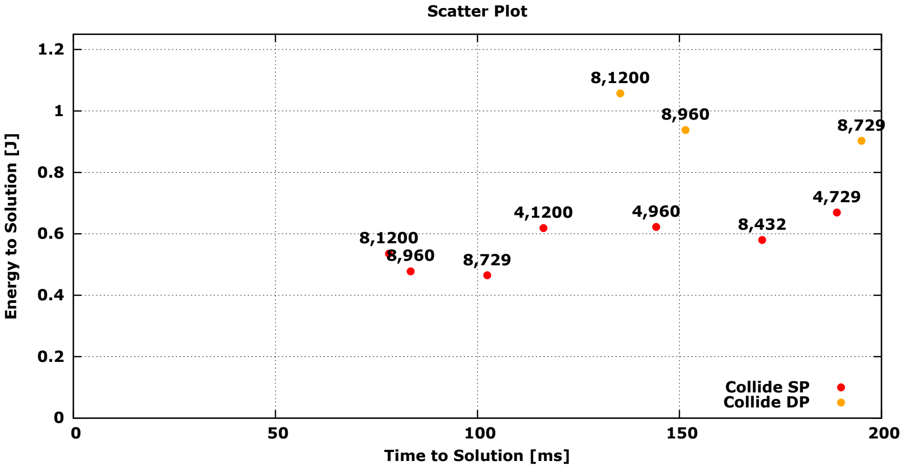
Energy to Solution vs Time to Solution (CPU) SP & DP



Energy/Time to Solution Propagate SP & DP

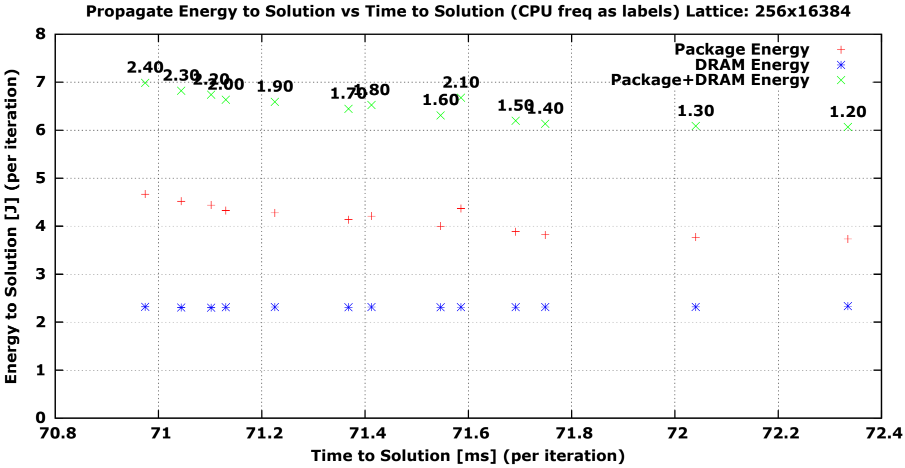


Energy/Time to Solution Collide SP & DP

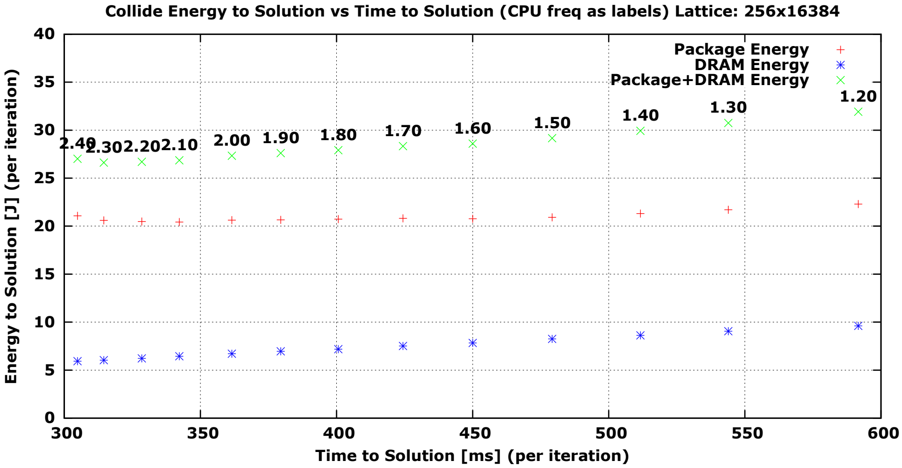


Outline

Energy/Time to Solution Propagate DP

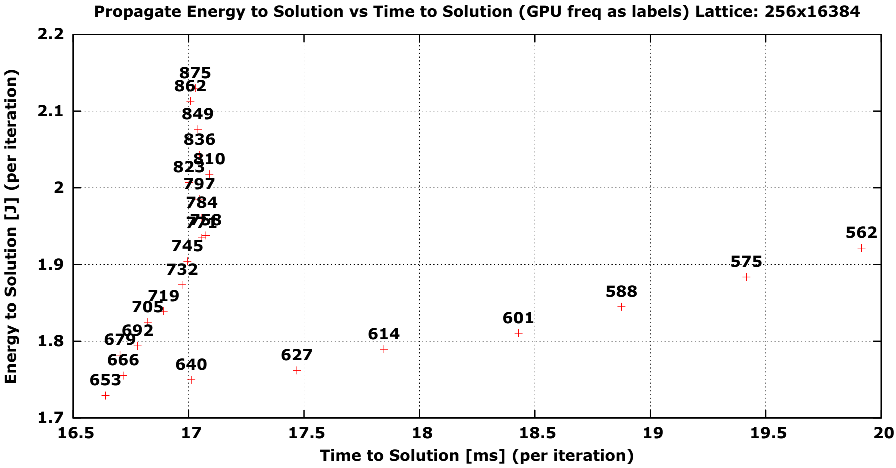


Energy/Time to Solution Collide DP

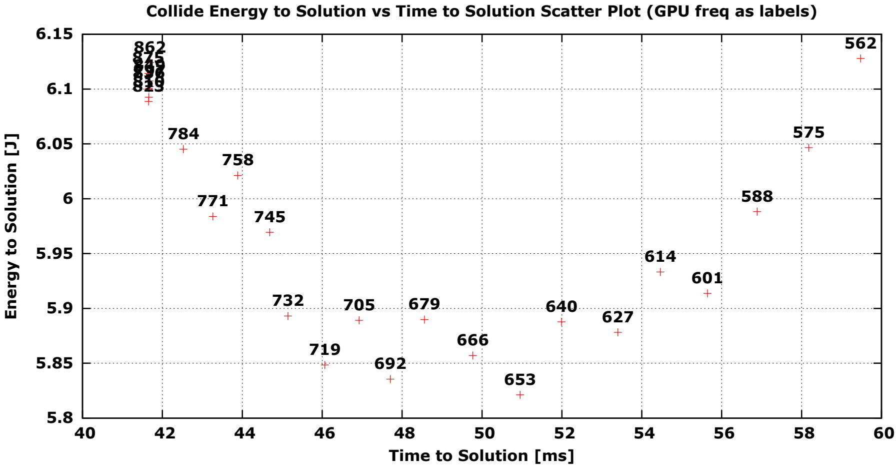


Outline

Energy/Time to Solution Propagate DP



Energy/Time to Solution Collide DP



Outline

Conclusions

- limited but not negligible power optimization is possible by adjusting clocks on a kernel-by-kernel basis (between $\approx 5 \cdots 25\%$).
- baseline power consumption (leakage current + ancillary electronics) is relevant (in particular for low-power processors $\approx 30\%$)
- options to run the processor at very low frequencies seem almost useless (at least for the adopted benchmark)

Future works

- perform similar measurements on different architectures (such as ThunderX ARM Processors)
- collect data for a fair comparison between architectures for several metrics
- investigate software tuning and multi-objective optimization techniques
- evaluate communication costs between different processors

Conclusions

- limited but not negligible power optimization is possible by adjusting clocks on a kernel-by-kernel basis (between $\approx 5 \cdots 25\%$).
- baseline power consumption (leakage current + ancillary electronics) is relevant (in particular for low-power processors $\approx 30\%$)
- options to run the processor at very low frequencies seem almost useless (at least for the adopted benchmark)

Future works

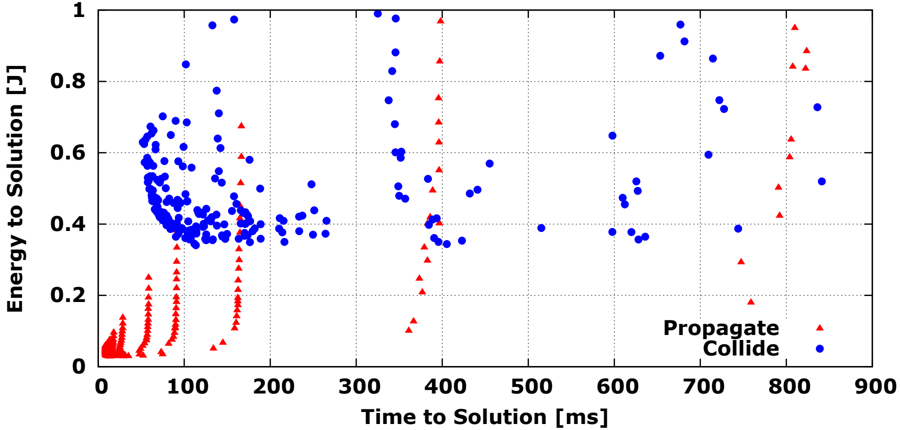
- perform similar measurements on different architectures (such as ThunderX ARM Processors)
- collect data for a fair comparison between architectures for several metrics
- investigate software tuning and multi-objective optimization techniques
- evaluate communication costs between different processors

Thanks for Your attention

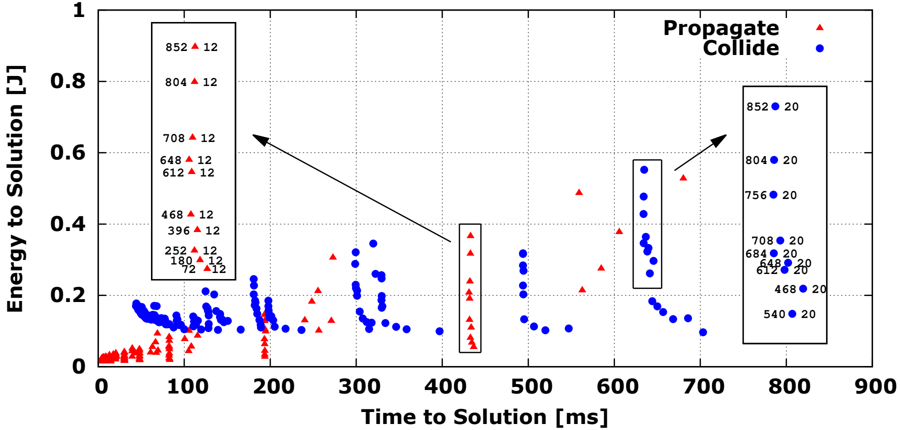
Processor	E_S [J] per iteration	T_S [ms] per iteration	EDP
NVIDIA GK20A	≈ 0.35	≈ 50	0.0175
ARM Cortex A15	≈ 0.75	≈ 50	0.0375
ARM Cortex A53	≈ 0.50	≈ 75	0.0375

Table: Collide Single Precision; lattice: 128x1024

Energy to Solution vs Time to Solution (CPU A15)



Energy to Solution vs Time to Solution (GPU GK20A)



Energy to Solution vs Time to Solution (GPU GK20A) zoom