Assessing performance of FeniCS on ARM architectures

Marois Vincent - Summer Internship 2017

UNIVERSITÉ DU LUXEMBOURG

FEniCS in short

- Open-source computing platform for solving partial differential equations
- Enables users to translate scientific models into finite element code
- Python & C++ interface
- Runs on a multitude of platforms ranging from laptops to high-performance clusters.
- Comes as a Docker image

Aim of the project

- Run FEniCS on an architecture with several constraints (size, power, memory…),
- Try to improve the performance of FEniCS using load-balancing,
- Retrieve sufficient data to compare the x86 & ARM platforms in the context of computational simulations with FEniCS.

Summary

- Why ARM ?
- Setup
- Recompiling FEniCS for ARM
- Cores combinations & energy consumption
- Load-balancing
- Weak-scaling benchmark
- Conclusion

Why ARM ? ARM vs x86

ARM	x86
Found mainly in mobile devices	Found in desktop PCs, laptops, servers, supercomputers
Reduced Instruction Set Computing (RISC)	Complex Instruction Set Computing (CISC)
Lower power consumption	Higher power consumption
(Relative) Lower performance	Higher performance
big.LITTLE = Heterogeneous computing	Same cores

January 2017 : "first ARM-based supercomputer called Isambard"

Setup : Odroid XU4

Samsung Exynos 5422 CPU (octacore ARM 32 bits)

2 Gbytes RAM

Power input : 4.8-5.2V / 0.8 -> 3.5A ~ 15W

Running Ubuntu Minimal

+ Wattmeter

Recompiling FEniCS for ARM

Creating a new Docker image :

Stable ('top image')
Dev-env
Base
Phusion base
multiarch/ubuntu-core:armhf-xenial

32 bits only, built on the odroid

Weak-scaling benchmark

Weak-scaling ?

Serial program solves problem of size P in time T

Weak scaling : Runs a larger problem. Solution size varies with fixed problem size per core

Strong scaling : Runs a problem faster. Solution size varies with fixed total problem size.

Test : solving Poisson equation in a unit cube mesh with 1 250 000 dofs.

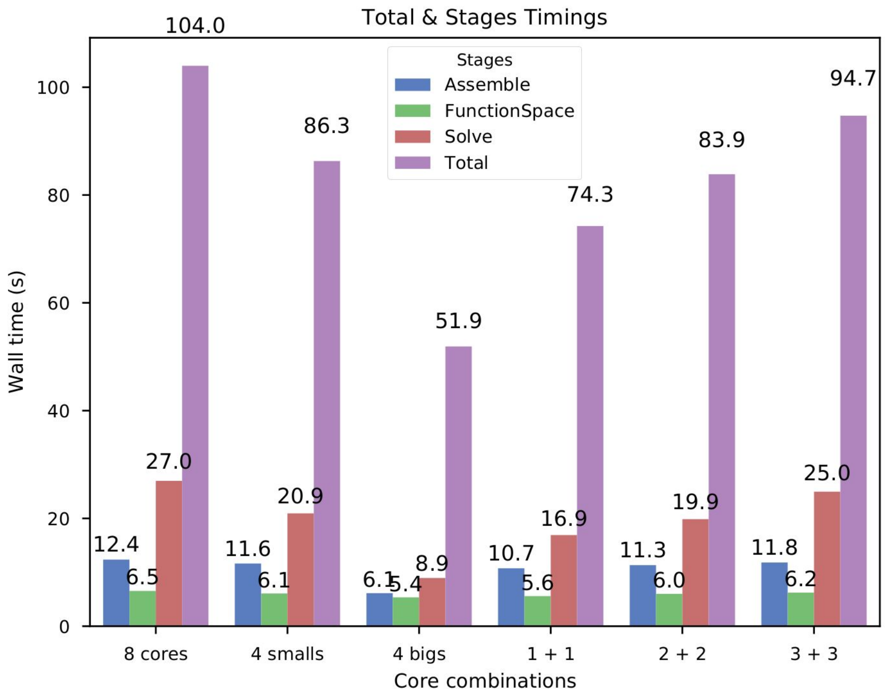
Using different cores combinations

4 “big” cores (Cortex-A15)

4 “small” cores (Cortex-A7)

Impact of combinations of both ?

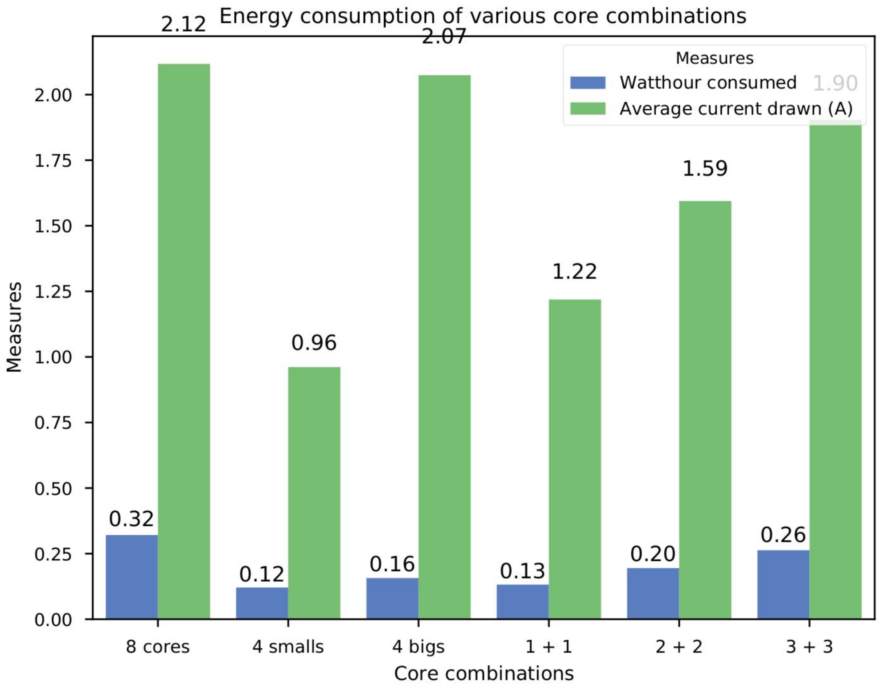
-> Using both simultaneously is not great : the big ones have to “wait” for the small ones to finish



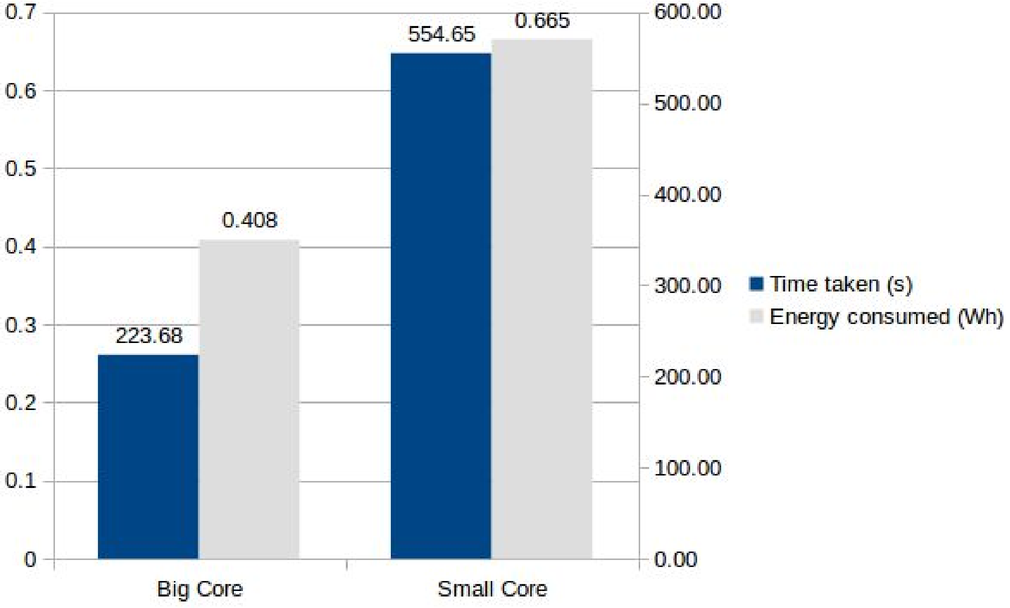
Using different cores combinations

-> 4 “big” cores draw 2x more current than the 4 “small” ones

Yet they consume about the same energy, as they are faster.



Using different cores combinations



Small core :

- 60% more energy consumed
- 2.5 x longer run time.

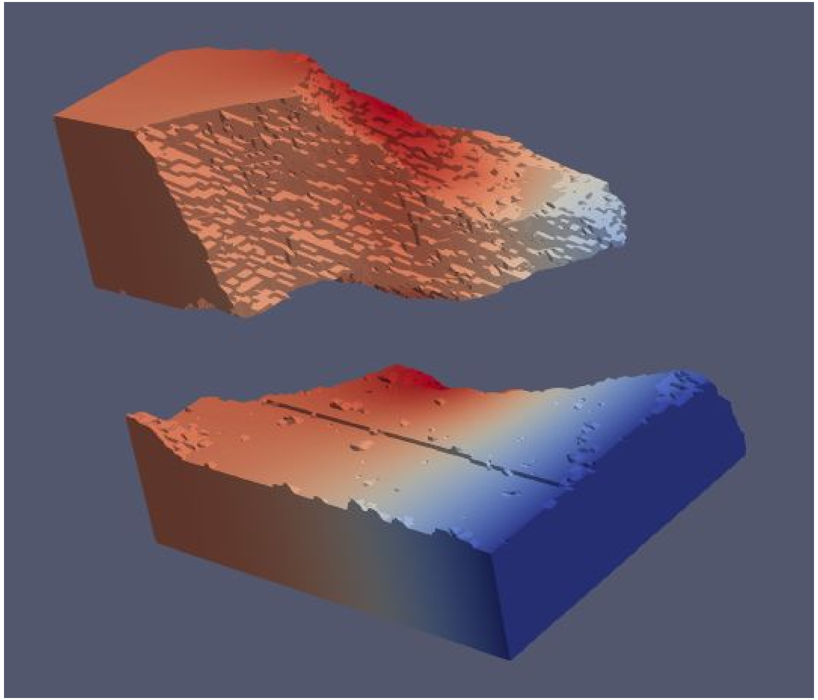
Load-balancing

How to improve performance when using all 8 cores ?

Weights related to mesh partitioning :

The mesh partitioner (ParMETIS) takes the whole mesh, and then splits it up into n parts

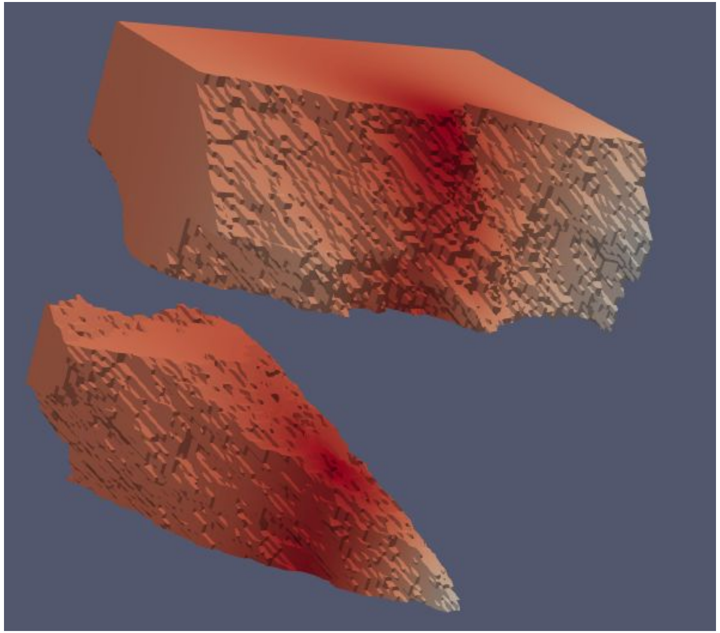
-> Modify these weights to put more load on “big” cores



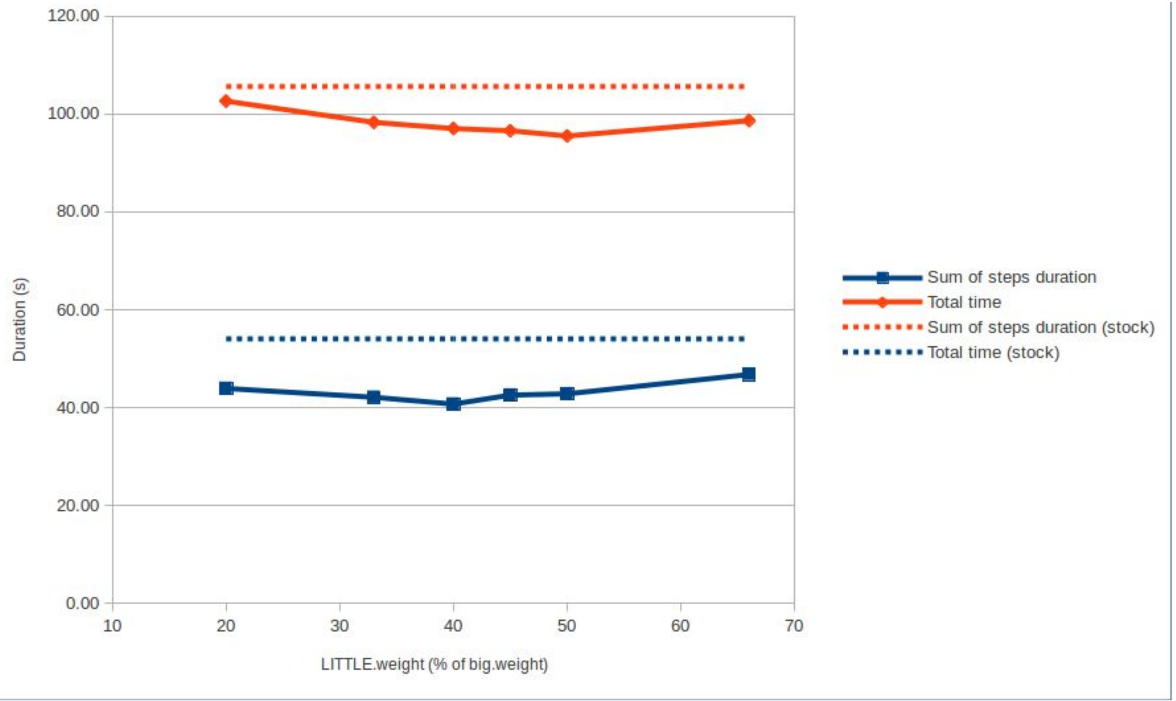
Load-balancing

Cells are no longer evenly distributed among cores

Partitions associated with “big” cores are bigger than the others

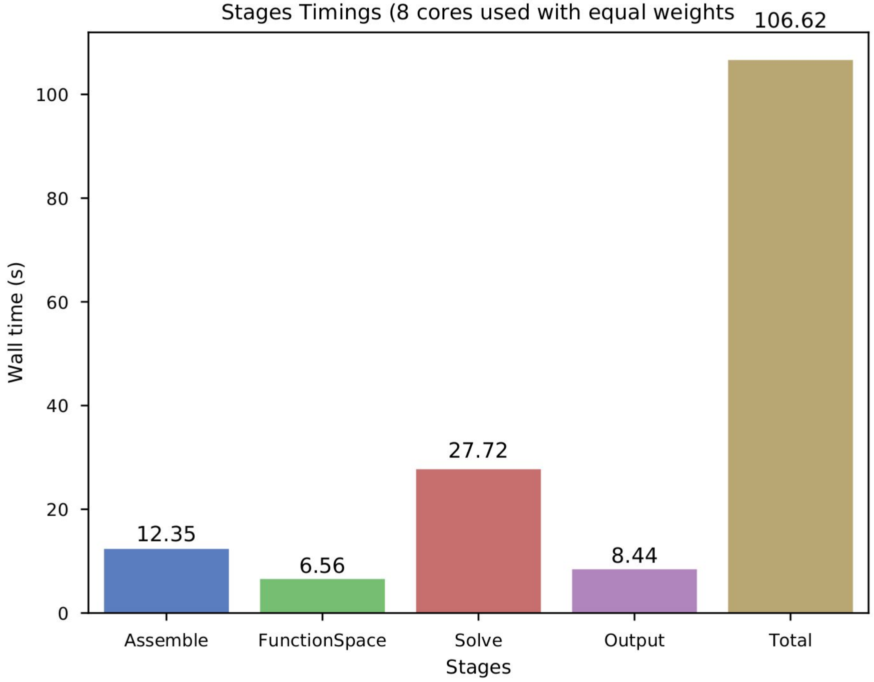


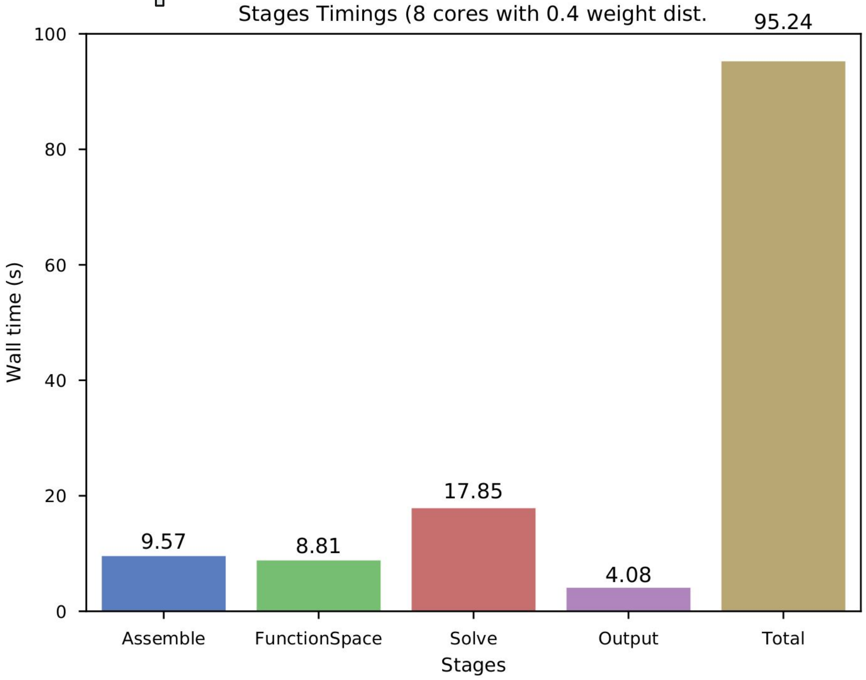
Load-balancing



Load-balancing

Not all steps are equal in terms of improvement

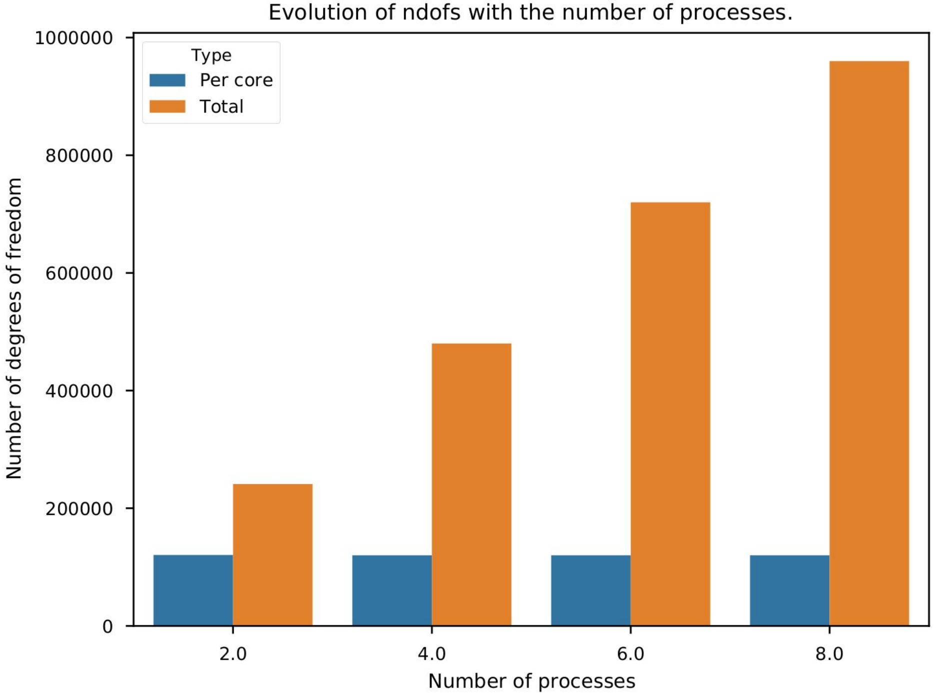




Weak-scaling benchmark

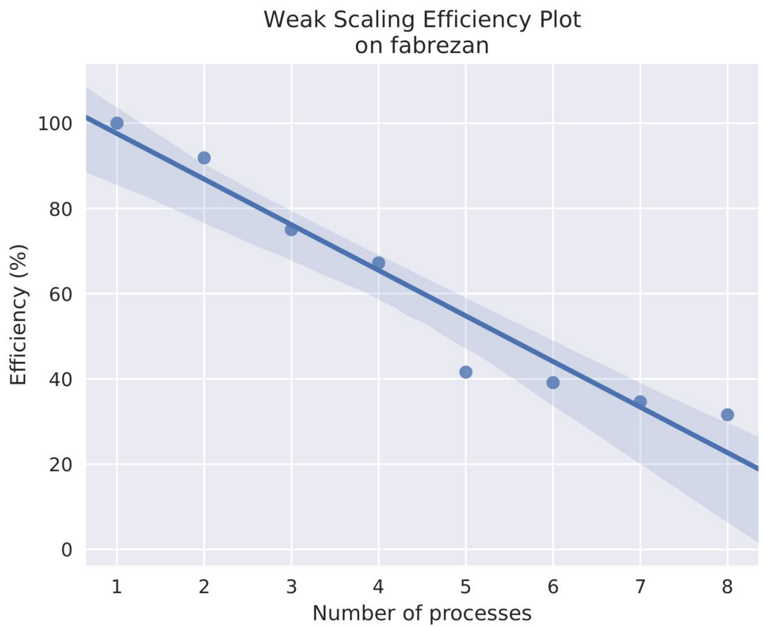
Weak-scaling is as intended :

The number of degrees of freedom per core doesn't vary, but the total one scales linearly



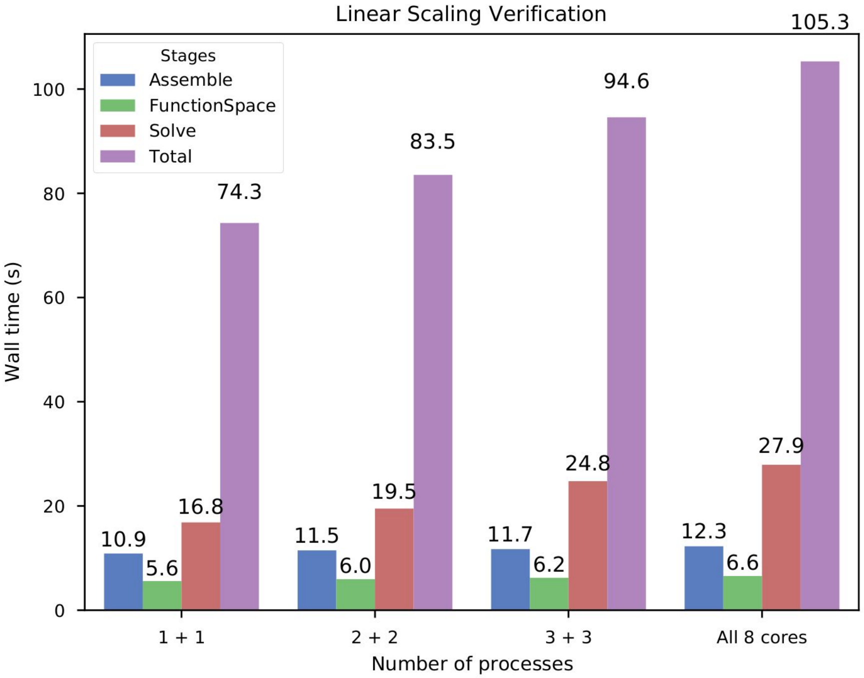
Weak-scaling benchmark

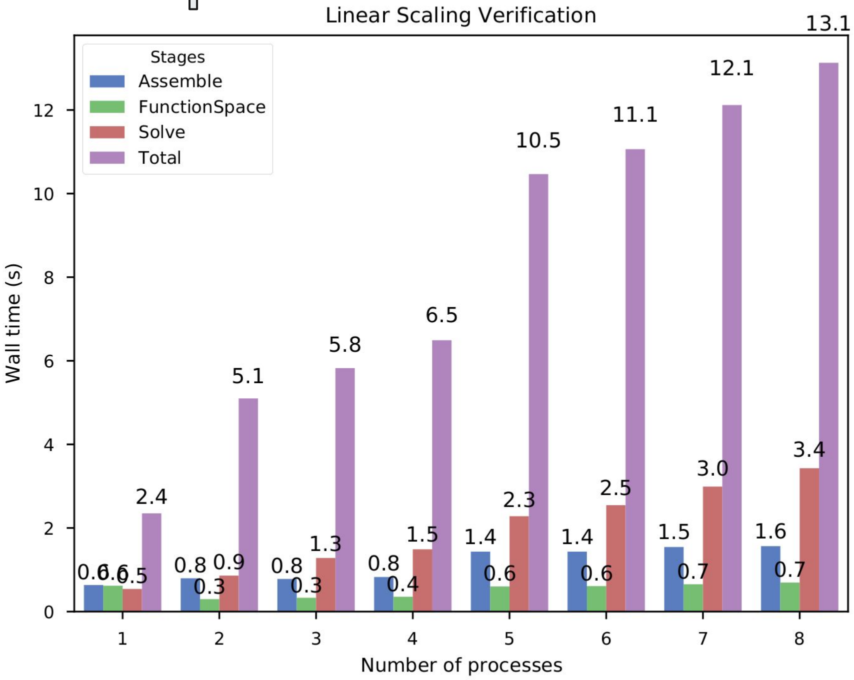
If the amount of time to complete a work unit with 1 processing element is **t1**, and the amount of time to complete **N** of the same work units with **N** processing elements is **tN**, the weak scaling efficiency (as a percentage of linear) is given as: **(t1 / tN) * 100%**



Weak-scaling benchmark

Average dissipated power > 65W





Conclusion

- FEniCS correctly runs on an ARM machine
- It is possible to take advantage of the different types of ARM cores to improve performance

Next steps :

- Use the Odroid I/O ports for a real-world application
- Improve performance further