Running the COSMO model on unusual hardware architectures - part 2

Davide Cesari

Arpae-SIMC, Bologna, Italy

1 Introduction

In a previous paper [1] it was shown how it is possible to run a complex numerical code such as the COSMO model on a small device designed for a completely different purpose: a home satellite TV receiver running the Linux operating system. In this paper a similar test is performed on an even smaller and cheaper –though more powerful– device, the Raspberry Pi single-board computer.

2 Characteristics of the device

The Raspberry Pi is a general-purpose computer on a very small board, measuring only $85x56mm^2$. It has a processor belonging to the ARM architecture, the one used by most of the smartphones today available on the market. The device used for the test is the Raspberry Pi 3 model B, the latest and most powerful model available at the moment, having a quad-core Broadcom processor with a GPU (Graphical Processing Unit) and 1 GB of memory. The board is also equipped, among the others, with wired and wireless network links, USB connections, video and audio output and SD card mass storage, which make it qualitatively comparable to an usual desktop or server computer.

This board is very popular among hobbyists for projects integrating external sensors and active devices with a powerful and easily programmable CPU, however, thanks to its computing power, it is perfectly suitable for traditional "number-crunching" applications. The official website is `http://www.raspberrypi.org`.

The most common operating system for the Raspberry Pi is a full version of Debian GNU-Linux, which, together with 1 GB of RAM, makes the question "is it possible to run the COSMO model on it?" superfluous.

The price of this board is around 35 Euros, thus making it one of the cheapest devices capable of running the COSMO model.

3 Preparation of the sequential test

In order to make a clean comparison with the results previously obtained, the same version of the compiler and of the COSMO model used in the previous tests, GNU gfortran 4.9.2 and COSMO version 5.00 respectively, have been used for the present work.

As shown in the previous paper on this subject, a viable way to produce an executable for such an architecture is *cross-compiling* on a desktop computer, i.e. generating the binary executable for the device on a computer having a different architecture and a special version of the compiler. This avoids the trouble of installing the complete compiler suite on the device and allows also to circumvent a possible unsuitability of the device to perform a full optimising compilation, e.g. due to lack of memory.

For compiling a sequential version of the COSMO code, the same instructions indicated in the previous paper have been followed. Due to the use of an ARM instead of a MIPS architecture on the device, the cross-compiler installation commands wee modified accordingly:

```
dpkg --add-architecture armhf
apt-get update
apt-get install crossbuild-essential-armhf
apt-get install gfortran-arm-linux-gnueabihf
```

After this step, the commands for compiling, linking and generating libraries for the Raspberry Pi are the usual commands such as `gfortran`, `gcc`, `ar`, etc. prefixed by the string `arm-linux-gnueabihf-`.

Also the setup of the sequential (single process, non-MPI) test case was the same used in the previous paper: a 3-dimensional idealised case of a rising warm bubble, implemented into COSMO by Ulrich Blahak [2], on a $21 \times 21 \times 40$ point grid with an horizontal step of 2km and a time step of 12s.

4 Performing the test

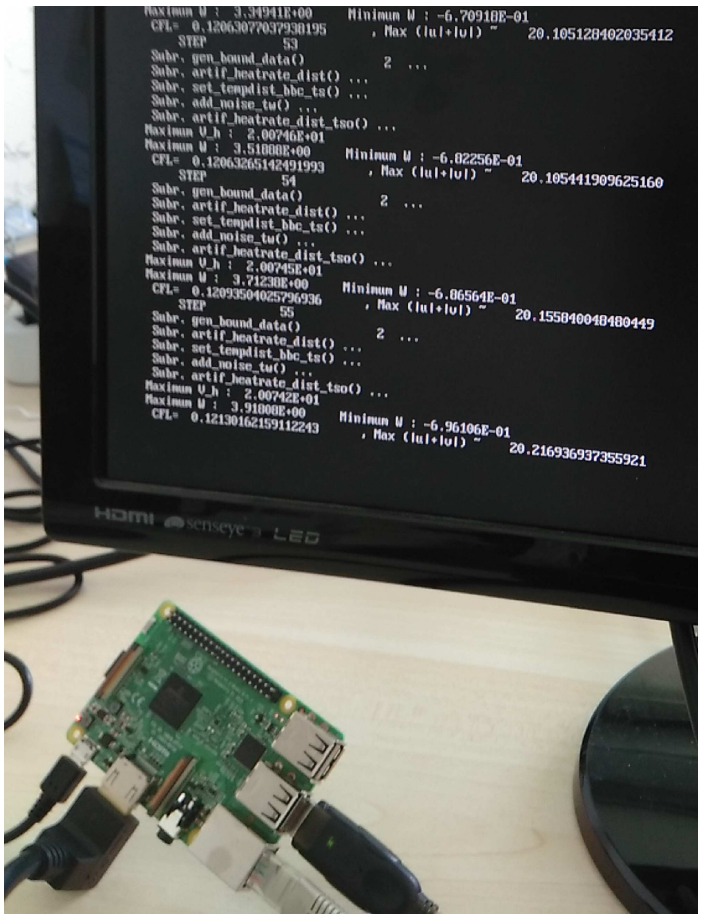

Figure 1: The Raspberry Pi connected to a huge screen, caught while running the COSMO model.

The test is performed by simply copying the executable and the namelists to the device connected to the network and by logging in to the device and running the COSMO model as usual. Since the Raspberry Pi, unlike the devices used in the previous paper, can have a console on the connected keyboard and monitor, the process of running the model on it can have a more exciting visual feedback on the screen as shown in the photo at figure 1.

5 Results of the sequential test

Table 1 summarises the results of the sequential test in terms of total wall-clock time required for one hour of forecast with the configuration described, as reported in the YUTIMING file. The table shows also the results obtained on the previously tested MIPS platforms as well as the results on a state of the art HPC computing node (price ≈2000 EUR) using a single processing core.

These results show that the Raspberry Pi lies logarithmically in the middle between the weak MIPS TV receiver tested in the previous work and the HPC computing node.

Platform	wall clock time (s)
Raspberry Pi 3B	139
Gigablue 800 UE	1111
Gigablue 800 SEplus	28649
HPC computing node	12

Table 1: Summary of the sequential tests performed, including the old results.

6 Parallel MPI tests

Since the Raspberry Pi has a processor with multiple computing cores and much more memory than the MIPS devices previously tested, a second and more interesting test with an MPI version of the code has been set up. This parallel version of the code can simultaneously run on two or more of the available cores and the parallel processes communicate through the shared memory.

The compilation of the MPI version of the COSMO model has also been performed as cross-compilation on an external host with a different architecture, this time generating a dynamical executable linking shared libraries. However, since the MPI software involves not just linking with additional libraries, but also a more complex compilation and runtime environment, the cross-compilation process did not work as cleanly as before, but it required some dirty tricks and hand corrections, so it is not described here.

Anyway, thanks to the relatively powerful hardware for the device under test and the availability of a complete operating system on it, it is perfectly feasible to compile the COSMO model with MPI support directly on the device, in the same way as it is usually compiled on a workstation or HPC login node.

Initially, the same test introduced before has been performed with the MPI version of the COSMO model, using from one to all of the four computing cores available. For comparison, the same test has been performed on the HPC node already used for the sequential test, using all the available processors/cores. The results are shown in table 2.

Platform	MPI processes and geometry	wall clock time (s)
Raspberry Pi 3B	1×1	138
Raspberry Pi 3B	1×2	98
Raspberry Pi 3B	1×3	89
Raspberry Pi 3B	1×4	92
Raspberry Pi 3B	2×2	99
HPC computing node	1×12	5.4

Table 2: Summary of the first parallel test performed.

This proves that the COSMO model with the setup described above shows some parallel scaling capability on the Raspberry Pi, but it can hardly profit of the third computing core, not counting the fourth.

It can also be noted that the MPI version does not introduce extra overhead with respect to the sequential (so-called "dummy MPI") version of the code, when run as a single MPI process.

Due to the partially unsatisfactory scaling, a more challenging setup has been prepared, by doubling the number of grid points on either direction ($41 \times 41 \times 40$) while keeping the same space resolution and time step. The temperature disturbance ("bubble") has been kept of the same size in the center of the enlarged domain.

The scaling results of this second experiment are shown in table 3.

Finally, another test, after further doubling the domain size on x and y directions, has been performed, whose results are shown in table 4.

These two tests show that with a more suitable domain size, the strong scaling of the COSMO code on the device under test is significantly better and all the four cores can give a positive contribution to the reduction of the time to solution.

Platform	MPI processes and geometry	wall clock time (s)
Raspberry Pi 3B	1×1	677
Raspberry Pi 3B	1×2	388
Raspberry Pi 3B	1×3	321
Raspberry Pi 3B	1×4	308
Raspberry Pi 3B	2×2	323
HPC computing node	1×12	15

Table 3: Summary of the second parallel test performed.

Platform	MPI processes and geometry	wall clock time (s)
Raspberry Pi 3B	1×1	3012
Raspberry Pi 3B	1×2	1713
Raspberry Pi 3B	1×3	1341
Raspberry Pi 3B	1×4	1230
HPC computing node	1×12	48

Table 4: Summary of the third parallel test performed.

7 Conclusions

Unlike the results presented in the previous paper, these results show that the architecture under test can compete with an HPC architecture in pure terms of performance per money and performance per watt.

Indeed the ratio between the figures for Raspberry Pi and a state of the art HPC node can be estimated to be approximately 1/60 for the price, 1/40 for the power consumption and 1/25 for the performance (of course referred to the COSMO model), thus with a little advantage for the Raspberry. Of course, due to the huge number of nodes that would be required, it is not feasible to employ such an architecture as it is for real parallel computing, but these results show that it is worth exploring this direction.

References

[1] Cesari, D., 2016: Running the COSMO model on unusual hardware architectures. COSMO Newsletter no.16 *Available online http://cosmo-model.org/content/model/documentation/newsLetters/newsLetter16/default.htm*

[2] Blahak, U., 2015: Simulating idealized cases with the COSMO-model. *Available online http://www.cosmo-model.org/content/model/documentation/core/artif_ docu.pdf*, 48pp.