Comparative study of embedded computing platforms using sequential programming

R. Ramírez-Rubio, E. Solórzano-Alor and M. Aldape-Pérez

Instituto Politécnico Nacional, IPN
Centro de Innovación y Desarrollo Tecnológico en Cómputo, CIDETEC
México D.F., México
rogelioramirezr@hotmail.com; eduardosolorzano22@hotmail.com;
maldape@gmail.com

Abstract. This paper presents a comparative study between three embedded computing and low-cost platforms. Computing platforms are 32-bit RISC based processors, ideal for low power applications. During the experimental phase, six numerical methods were implemented in C++ language for performance comparison among them. The operating system used was Linux, with a specific distribution for each platform. Experimental results show that the Dual-Core ARM Cortex-A7 processor based platform shows a slightly higher performance regarding to Cortex-A9 Dual-Core processor based platform and ARM Cortex-A7 platform.

Keywords: Embedded computing, ARM processor, Numerical methods, Parallella, Cubieboard, Raspberry

1 Introduction

Today the computer systems are everywhere. It should not surprise that millions of computer equipment will be built each year to be used as desktops, laptops, workstations and servers. The amazing thing is that from the transistor miniaturization, many of the computer systems have electronic devices highly integrated dedicated to the execution of specific functions [4]. Embedded systems are found in a wide variety of everyday devices such as consumer electronics (phones, tablets, digital cameras, calculators and audio players) [1], for domestic use (microwave ovens, automatic answering, smart thermostats, video surveillance systems and lighting systems) [9, 13], office equipment (projectors, copiers, printers, scanners and alarm systems)[7] and automotive (control of transmission, anti lock brakes, fuel injection and active suspension)[11]. An embedded system typically costs much less than a team of computing and have specific characteristics that distinguish between them [12]. Fixed function systems: Such systems are usually inexpensive and are designed for repeatedly execute a specific program [5]. Restricted systems: Such systems are designed under metric (restrictions) high performance with low power consumption, and using semiconductor materials highly integrated [8]. Reactive and real-time systems: Such systems must react to the changing environment with the least possible delay;

the metric governing the design of such systems is the measurement of variables and decision making in real time [10]. Currently there are embedded systems developed under the paradigm of Single Board Computer (SBC, for its acronym) that promote the teaching of Computer Science [3]. This paper presents a comparative study of the performance of three low-cost embedded systems, which have been widely used in the development of everyday applications. The rest of the paper is organized as follows: In Section 2, are presented the characteristics of the platforms of experimentation, in Section 3, a brief description of the numerical methods that were used for the experimental phase Section 4, in Section 5 are presented results and finally in Section 6, some conclusions.

2 Development platforms

2.1 Parallella

Parallella is a high performance computing device, which can be used as a standalone computer, such as an embedded system or as part of a cluster. The Parallella platform includes a dual core ARM A9 processor low power of consumption, which is able to work with different Linux distributions, this makes the platform be attractive for its wide versatility with working environments, which besides being accessible, give you ease of use to the user. One of the applications in which can be used the Parallella platform is processing and image analysis, a clear example is the face detection, this issue has been extensively studied with low-cost computing and innovative algorithms [14].

2.2 Cubieboard

Cubieboard is a single computer , open source produced by company Cubietech, based in Zhuhai, China. It has a good performance mainly in an tasks of office, games and entertainment, thanks a with its A20 processor.

The A20 processor is based on a dual core ARM architecture Cortex-A7 and integrates a GPU ARM Mali-400 MP2, delivering good performance and a reliable system performance, plus compatibility with different games. The A20 processor supports 2160p video decoding and encoding of 1048p, this makes the Cubieboard a good solution for mobile or desktop applications. Due to capacity, relatively small size compared to other cluster more robust and low cost of platform Cubieboard has been used for processing databases [15].

2.3 Raspberry

Raspberry Pi Model B is a credit-card sized computer of low cost, that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is capable of doing everything you would expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games all this with an 700 MHz ARM 1176JZF-S processor and 512 MB RAM. The Raspberry Pi B has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. Also implementing multi-label classification algorithm with low cost. With the facilities available in Mathematica software for Raspberry Pi, the line of code required for implementing data mining algorithms can be reduced sufficiently [6].

Below, in Table 1 shown relevant characteristics of system the platforms to evaluate :

Table 1: Comparative table about the characteristics of the target platforms

Platform / Features	Parallella	Cubieboard	Raspberry
Processor	Xilinx Zynq Dual-core ARM A9 XC7Z020	AllWinnerTech SOC A20 Dual-Core ARM A7	700 MHz ARM 1176JZF-S
RAM	1 GB DDR3	1 GB DDR3	512 MB DDR3
Storage	Micro-SD memory	3.4 GB NAND flash or Micro-SD memory	Full-size SD-Memory

3 Numerical methods

The numerical methods are essential to approximate analytical values so difficult or impossible to obtain. One of the basic problems in the numerical approach is the search for roots of a function, that is, when the value of x-axis value of the ordinate is zero or very close, that is below a threshold determined so that is searched f (x) = 0. The numerical methods listed below are used to obtain an approximation to the root of certain functions. It should be mentioned that each method has advantages and drawbacks, which include convergence capacity, types of function where the method can never converge, among others. Finally, numerical methods can be used even if the objective function can be treated in an analytical form allowing the use of computer systems. The numerical methods listed below are used to find roots of a function, using a classical reference [2].

3.1 Bisection

This method is based on the intermediate value theorem, also it is known as binary search method. It has a function f that is continuous on the interval [a, b] with $f(a)$ and $f(b)$ of different signs, ie $f(a) * f(b) < 0$. According to the intermediate value theorem, there is a value in p (a, b) such that $f(p) = 0$. Although the method is applied even if there more a root in the range, for simplicity it is assumed that the root of the interval is unique. The method requires split several times by half subintervals of [a, b], and in each step, locating the half containing a p.

To begin, suppose that $a_1 = a$ and $b_1 = b$, and let p_1 the midpoint of [a, b]; ie,

$$p_1 = a_1 + (b_1 - a_1)/2 = (a_1 + b_1)/2 \tag{1}$$

If $f(p_1)$, then $p = p_1$; if not so, then $f(p_1)$ has the same sign to $f(a)$ or $f(b_1)$. If $f(p_1)$ and $f(a_1)$ have the same sign, then $p \in (p_1, b_1)$ and took $a_2 = p_1$ and $b_2 = b_1$. If $f(p_1)$ and $f(a)$ have opposite signs, then $p \in (a_1, p_1)$ and took $a_2 = a_1$ and $b_2 = P1$. Then we apply the process to the interval $[a_2, b_2]$.

3.2 Newton - Raphson

This method, also known as Newton - Raphson, is one of the numerical techniques to solve a problem of search of roots $f(x) = 0$ more powerful and popular. You can be viewed in different ways, one of which is the possibility of deriving a technique that allows for a faster convergence than that offered by other types of functional iteration.

It begins with an initial approximation p_0 to calculate

$$p = p_0 - f(p_0)/f'(p_0) \tag{2}$$

and if is checked

$$|p - p_0| < tolerance \tag{3}$$

not be redefined

$$p_0 = p \tag{4}$$

and is iterated again.

3.3 Müller

This method is an extension of the secant method, the method of Müller uses three approaches x_0, x_1 and x_2 and determine the next approximation x_3 to consider the intersection of the x axis with the parabola passing through $(x_0, f(x_0))$, $(x_1, f(x_1))$ and $(x_2, f(x_2))$.

3.4 Secant

This method starts with two initial guesses p_0 and p_1, the approximation p_2 is the intersection of the axis line connecting x and $(p_0, f(p_0))$ and $(p_1, f(p_1))$. The approximation p_3 is the intersection of the axis line connecting x and $(p_1, f(p_1))$ and $(p_2, f(p_2))$, and so on. The method must iterate until they get $|p - p1| < tolerance$ and each iteration should be replaced with the values to be treated, that is $p_0 = P_1, p_1 = p$, etc .

3.5 Fixed Point

A fixed point of a function g is a number p for which $g(p) = p$. Given a problem to find a root $f(p) = 0$, you can define a function g with a fixed point p in various ways; for example, like $g(x) = x - f(x)$ or $g(x) = x + 3f(x)$. Conversely, if the function g has a fixed point in p, then the function defined by $f(x) = x - g(x)$ has a zero at p.

3.6 Steffensen

This method can be considered as a combination of fixed point and Aitken methods. As the Aitken method essentially accelerates the convergence of other method, this method can be defined as accelerated fixed point method. This method has a fast convergence and does not require, as in the case of Newton's method, the evaluation of a derivative. Moreover, it has the additional advantage that the iteration process only needs a starting point. Then, it is found a $p = g(p)$ given an initial approximation p_0.

It must be calculated in each iteration:

$$p_1 = g(p_0) \tag{5}$$

$$p_2 = g(p_1) \tag{6}$$

$$p = p_0 - ((p_1 - p_0)^2)/(p_2 - 2p_1 + p_0) \tag{7}$$

and must iterate until $|p - p_0| < tolerancia$.

4 Experimental phase

In this phase, tests were performed with the above numerical methods programmed in C++ language, where C++ is a language created for the extension of C that allows the manipulation of objects, besides being a compiled language.

The tests that were performed consisted in evaluating the following expressions (objective functions), development platforms and programming language C++, in all of them a plain text editor and the GNU g++ compiler were used.

Objective Functions

$$f_1(x) = x^3 + x - 6 \tag{8}$$

$$f_2(x) = x^3 + 4x^2 - 10 \tag{9}$$

$$f_3(x) = 8x - cos(x) - 2x^2 \tag{10}$$

The source code is compiled and executed on a Linux distribution; Cubieez for Cubieboard, Linaro for Parallella and Raspbian for Raspberry platform, in each platform shows execution times of programs, with the ”time” command in Linux, thus obtaining the Table 2, Table 3 and Table 4.

The Table 2 shows the execution times each of numerical methods programming in C++ language on the platform Parallella.

Table 2: Execution time (seconds) on Parallella with C++.

Function / Method Numeric	$f_1(x)$	$f_2(x)$	$f_3(x)$
Bisection	0.018	0.018	0.018
Newton - Rapshon	0.021	0.028	0.021
Müller	0.020	0.019	0.019
Secant	0.018	0.018	0.021
Fixed Point	0.018	0.018	0.018
Steffensen	0.018	0.021	0.018

As can be seen in the Table 3, showing the execution times of numerical methods implemented Cubieboard.

Table 3: Execution time (seconds) on Cubieboard with C++.

Function / Method Numeric	$f_1(x)$	$f_2(x)$	$f_3(x)$
Bisection	0.010	0.010	0.018
Newton - Rapshon	0.010	0.019	0.010
Müller	0.010	0.010	0.010
Secant	0.010	0.010	0.009
Fixed Point	0.010	0.009	0.009
Steffensen	0.010	0.009	0.010

The Table 4 shows the runtimes in Raspberry platform with numerical methods programmed in C ++ language.

Table 4: Execution time (seconds) on Raspberry with C++.

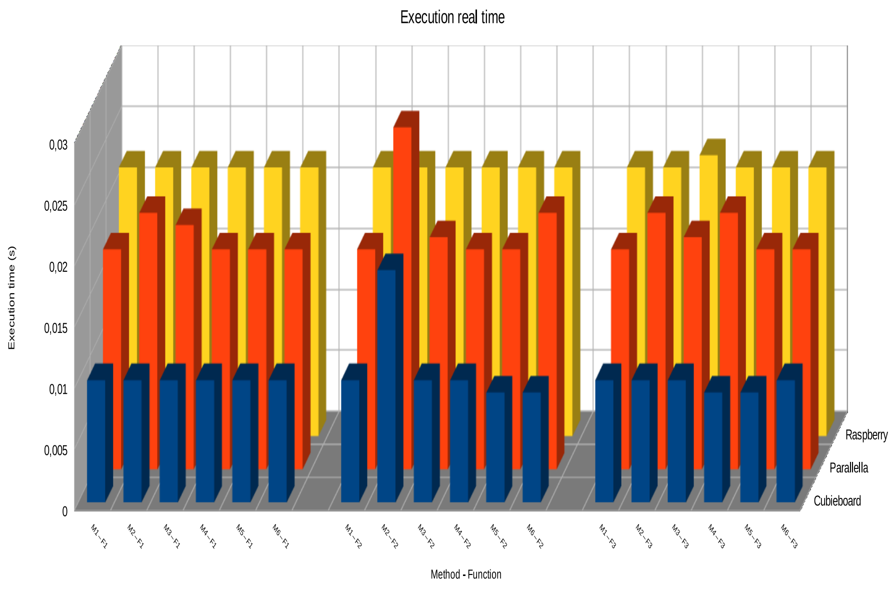
Function / Method Numeric	$f_1(x)$	$f_2(x)$	$f_3(x)$
Bisection	0.022	0.022	0.022
Newton - Rapshon	0.022	0.022	0.022
Müller	0.022	0.022	0.023
Secant	0.022	0.022	0.022
Fixed Point	0.022	0.022	0.022
Steffensen	0.022	0.022	0.022

5 Results and Conclusions

In this section the results from previous tables are presented graphically to enhance the understanding of the information.

Fig. 1 shows Parallella, Cubieboard and Raspberry execution times with numerical methods programmed in the C ++ language.

Fig. 1: Chart of execution times in the three platforms with C++ language.



In the chart they are shown the execution time of the numerical methods, where M1 is bisection, M2 is Newton - Rapshon, M3 is Müller, M4 is Secant, M5 is Fixed Point, M6 is Steffensen and function (8), (9) and (10) are the objective functions respectively.

The Fig. 1 shows that the real time are kept steady except for the method of Newton - Rapshon presented unstable behavior to resolve the function (9) in Parallella and Cubieboard. Based on the results shown in Figure 1, the methods implemented in the C++ language, comparing the performance time Parallella with the performance of Cubieboard can see that in every one of the methods performance was better in Cubieboard as seen by comparing in the chart, the Raspberry showed a slightly lower performance compared to the other two platforms, though more stable in the times of execution.

According to the results, we can conclude that the Cubieboard platform is slightly best in the execution of algorithms of numerical methods, so it is convenient to use this system embedded in numerical applications for higher speed in the execution of numerical algorithms.

Acknowledgments

The authors of the present paper would like to thank the following institutions for their economical support to develop this work: Science and Technology National Council of Mexico (CONACYT), SNI, National Polytechnic Institute of Mexico (COFAA, SIP, CIDETEC, and CIC).

References

1. Ivan Cibrario Bertolotti and Tingting Hu. Modular design of an open-source, networked embedded system. *Computer Standards & Interfaces*, 37(0):41 – 52, 2015.
2. R.L. Burden and J.D. Faires. *Análisis numérico*. International Thomson Editores, 2002.
3. Edward T.-H. Chu and Chi-Wei Fang. Calee: A computer-assisted learning system for embedded {OS} laboratory exercises. *Computers & Education*, 84(0):36 – 48, 2015.
4. Xiaocong Fan. Chapter 1 - introduction to embedded and real-time systems. In Xiaocong Fan, editor, *Real-Time Embedded Systems*, pages 3 – 13. Newnes, Oxford, 2015.
5. Tetsuo Furuichi and Kunihiro Yamada. Next generation of embedded system on cloud computing. *Procedia Computer Science*, 35(0):1605 – 1614, 2014. Knowledge-Based and Intelligent Information &amp; Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings.
6. Neethu John, R. Surya, R. Ashwini, S. Sachin Kumar, and K.P. Soman. A low cost implementation of multi-label classification algorithm using mathematica on raspberry pi. *Procedia Computer Science*, 46(0):306 – 313, 2015. Proceedings of the International Conference on Information and Communication Technologies, {ICICT} 2014, 3-5 December 2014 at Bolgatty Palace &amp; Island Resort, Kochi, India.

7. Hau T. Ngo, Robert W. Ives, Ryan N. Rakvic, and Randy P. Broussard. Real-time video surveillance on an embedded, programmable platform. *Microprocessors and Microsystems*, 37(6-7):562 – 571, 2013.
8. Jakub Novak and Petr Chalupa. Implementation of mixed-integer programming on embedded system. *Procedia Engineering*, 100(0):1649 – 1656, 2015. 25th {DAAAM} International Symposium on Intelligent Manufacturing and Automation, 2014.
9. Ashish Pandharipande and David Caicedo. Daylight integrated illumination control of {LED} systems based on enhanced presence sensing. *Energy and Buildings*, 43(4):944 – 950, 2011.
10. Jigneshkumar J. Patel, Nagaraj Reddy, Praveena Kumari, Rachana Rajpal, Harshad Pujara, R. Jha, and Praveen Kalappurakkal. Embedded linux platform for data acquisition systems. *Fusion Engineering and Design*, 89(5):684 – 688, 2014. Proceedings of the 9th {IAEA} Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.
11. R. Pons, A. Subias, and L. Trave-Massuyes. Iterative hybrid causal model based diagnosis: Application to automotive embedded functions. *Engineering Applications of Artificial Intelligence*, 37(0):319 – 335, 2015.
12. R.A. Shafik, A. Das, S. Yang, G. Merrett, and B.M. Al-Hashimi. 9 - design considerations for reliable embedded systems. In Jonathan Swingler, editor, *Reliability Characterisation of Electrical and Electronic Systems*, pages 169 – 194. Woodhead Publishing, Oxford, 2015.
13. Peter L. Venetianer and Hongli Deng. Performance evaluation of an intelligent video surveillance system - a case study. *Computer Vision and Image Understanding*, 114(11):1292 – 1302, 2010. Special issue on Embedded Vision.
14. Ming Yang, James Crenshaw, Bruce Augustine, Russell Mareachen, and Ying Wu. Adaboost-based face detection for embedded systems. *Computer Vision and Image Understanding*, 114(11):1116 – 1125, 2010. Special issue on Embedded Vision.
15. Ni Zhang, Yu Yan, Shengyao Xu, and Wencong Su. A distributed data storage and processing framework for next-generation residential distribution systems. *Electric Power Systems Research*, 116(0):174 – 181, 2014.