

Parallel Programming on Embedded Multicore System ESP32

Universitatea Politehnica Timișoara



Marian Belean
Franz Joseph Pal

WS2019/2020
October 29, 2019

Abstract

The following documentation will focus on the principles of parallel programming in general and the mathematical background. In addition to the different parallel programming architectures, the various models for their implementation are also discussed. Moreover, in this thesis the prerequisites for mathematical calculation models, which are suitable for Parallel Programming, are elaborated.

For a practical example, the ESP32 microcontroller was chosen, an embedded multicore system. After a brief introduction to the hardware itself, further details of the project structure and the development of the application will be presented. Therefore, a short example will be explained to focus on the basics of parallel programming.

Finally, the aim of the project and the documentation is an automatic benchmark setup and a webfrontend result overview for visualization purposes, which will be discussed in more detail in the conclusion.

Declaration

I hereby certify that I have done the final thesis on my own, that I have completely and accurately stated all the aids I have used and identified everything individually, which was taken from the work of others unchanged or with modifications.

The topic of the submitted work was jointly with Mr. / Mrs. (...) (Bachelor and Master Thesis No. (...)).

Timișoara, the October 29, 2019

Signature:

Non-disclosure notice

This work contains confidential information. In spite of the anonymous presentation of the researched organisations, readers might conclude their identity. Therefore copying, quoting or publishing is not allowed without my explicit authorisation. Furthermore, disclosure of the information to anyone other than the examination board or lecturers is not authorized.

Contents

1	Introduction	1
2	Overview	2
2.1	Problem definition	2
2.2	Objective of the documentation	2
3	Parallel Programming in General	3
3.1	Basic Concept	3
3.1.1	Principles of Parallel Computing	5
3.2	Definition of parallel mathematical computations	6
3.3	Parallel Computer Architecture	7
3.3.1	Flynn's Taxonomy of Parallel Architectures	7
3.3.2	Thread Level Parallism	7
3.4	Parallel Programming Models	7
3.4.1	Classification of Parallel Programming Models	8
4	Project documentation	9
4.1	Concept development	9
4.2	Project structure	10
4.3	Simple mathematical computation examples for Parallel Programming	11
4.4	Class diagramm	12
4.4.1	C++ Backend benchmark	12
4.4.2	Vuejs Frontend	12
4.5	Benchmark setup	13
5	Conclusion	14
A	Additional documents	15

Chapter 1

Introduction

Multicore systems are becoming increasingly popular as part of digitization and Industry 4.0¹ (German/EU) [2] [or 1] - also known as smart manufacturing in the USA [see 29, p1] - and are playing an important role in data processing and process automation [see 33, p294] [or 28, p1]. On the other hand, in addition to efficiency in energy consumption, performance in terms of computation time [see 33, p294] is required in almost every application field of multicore systems.

In fact, multicore hardware is not only expecially for smart manufacturing. Nowadays in almost every smart application like smart phones², wearables³ or home automation we can find multicore embedded hardware platforms, which garantued high performance [see 4, p7], network connectivity, security and reliability [see 36, p5]. This field of application is also known as Internet of Things (IoT)⁴.

Especially for embedded systems mathematical models as well as numerical solutions, which can be executed both simply and parallel, are suitable. The question arises to what extent parallel execution of different sub-tasks to calculate a problem [see 32, p4] increases the desired cost factor in terms of energy consumption [see 13, chapter 3] and computational efficiency [see 32, p4 Figure 3].

¹add.: <https://www.epicor.com/en-ae/resource-center/articles/what-is-industry-4-0/>

²e.g. ARM based processors for mobile phones like <https://www.arm.com/solutions/mobile-computing/smartphones>

³add. information on ARM based solutions and the current trend in wearables: <https://www.arm.com/solutions/wearables>

⁴for additional information about Internet of Things, please see [20]

Chapter 2

Overview

2.1 Problem definition

Compared to single-core execution of tasks, multi-core embedded hardware platforms like the ESP32¹ provide the ability to develop advanced parallel computing software applications to reduce execution time and power consumption.

On the one hand, a major problem is choosing the right hardware platform to meet the cost and size factor, and moreover, whether a single-core or multi-core calculation is required. Therefore, context switching time, power consumption and total execution time must be included in the evaluation.

In order to develop an optimal solution, the hardware platform must be included in addition to the mathematical model of the problem itself. So in this case, suitable prerequisites and characteristics can be worked out in order to make an evaluation of "Parallel Computation Tasks on Embedded Multicore Systems" possible.

2.2 Objective of the documentation

The main goal of this documentation is to focus on the current parallel programming techniques, depending on the execution time in general and the required mathematical model. For this purpose, an application which can compute different sections of the Mandelbrot fractal [see 19, p11] will be developed to compare single core and multi-core calculations. Before the practical implementation, an investigation based on parallel architectures and programming models will be conducted.

The elaboration is divided into three different chapters: In the first chapter, the results of the general research are presented [see Chapter 3]. After that, the second chapter is pointing out the practical implementation of the developed application on the ESP32 [see Chapter 4]. In the end, the results including the webforntend and the automatic benchmark setup [see Chapter 5] will be discussed.

¹add. information: <https://www.espressif.com/en/products/hardware/socs>

Chapter 3

Parallel Programming in General

3.1 Basic Concept

Amdahl's law [5]:

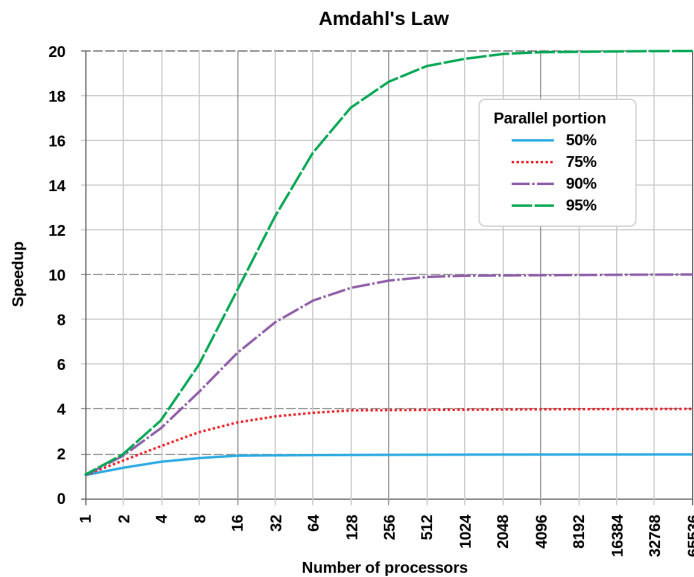


Figure 3.1: ...

- “the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers” [see 8, p80]; can be transferred on single and multi-core processors or even on multithreading, but in fact, like Amdahl claimed too, addressing hardware, and nowadays switching context time was not considered here! Valiant noted in 1990, “no substantial impediments to general-purpose parallel computation” exist [see 8, p85], though there are limits, as shown [see in Sec. 10.8, p85].

- “the fraction of the computational load... associated with data management housekeeping ... accounts for 40% of the executed instructions”, “this overhead appears to be sequential so that it is unlikely to be amenable to parallel processing techniques” [see 8, p80]; amount of overhead (data management and addressing based on hardware restrictions), which can not be parallelized and reduces the factor of speed increase after parallelization. According to Amdahl, this factor can be reduced, but not with “parallel processing techniques”, probably with more precisely and efficient hardware.
- a resisting “upper limit on speedup [exists] and therefore, apart from the sequential fraction (the so called non parallelizable overhead), the remaining computations are perfectly parallelizable” [see 8, p81]
- the law in general: Let t_1 be the time taken by one processor solving a computational problem and t_p be the time taken by p processors solving the same problem. Finally let us denote the supposed inherently sequential fraction of instructions by f . Then, according to Amdahl, $t_p = t_1 (f + (1-f)/p)$ and the speedup obtainable by p processors can be expressed as:

$$\frac{t_1}{t_p} = \frac{1}{f + (1-f)/p}$$

3.1.1 Principles of Parallel Computing

Emphasises design: Amdahl's law highlights the pitfalls of looking for sticking-plaster speed-ups in serial programs – design for concurrency [see 23, p4]

Aim of concurrency and their effects on program structure or implementation [see 23, p5]:

- Flexibility: Environments will be more heterogeneous.
- Efficiency: parallel for a speed-up purposes, more pitfalls (memory latency, thread overheads etc.)
- Simplicity: Parallel codes will be more complicated. All the more reason to strive for maintainable, understandable programs.

3.2 Definition of parallel mathematical computations

...

3.3 Parallel Computer Architecture

...[see 30, p9]

...examples of parallelism [see 30, p11]

3.3.1 Flynn's Taxonomy of Parallel Architectures

...[see 11, p5]

...[see 30, p13]

...[see 10, p4]

...[see 5, p2]

3.3.2 Thread Level Parallism

...[see 30, p24]

3.4 Parallel Programming Models

Steps to evaluate a proper parallel design [see 23, p6]:

1. Finding Concurrency
2. Algorithm Structure
3. Supporting Structures
4. Implementation Mechanisms

3.4.1 Classification of Parallel Programming Models

3.4.1.1 Process Interaction

...[see 11, p4]

3.4.1.2 Problem decomposition

...[see 30, p105 ff.]

Chapter 4

Project documentation

4.1 Concept development

...

4.2 Project structure

...

4.3 Simple mathematical computation examples for Parallel Programming

...

4.4 Class diagramm

...

4.4.1 C++ Backend benchmark

...

4.4.2 Vuejs Frontend

...

4.5 Benchmark setup

...

Chapter 5

Conclusion

...

Appendix A

Additional documents

Bibliography

- [1] Plattform Industrie 4.0. “Positionspapier, Leitbild 2030 für Industrie 4.0 - Digitale Ökosysteme global gestalten”. In: *Plattform Industrie 4.0* (Apr. 2019). URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Positionspapier%20Leitbild.pdf?__blob=publicationFile&v=5.
- [2] Plattform Industrie 4.0. *Was is Industrie 4.0?* 2019. URL: <https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html>.
- [3] Umut A. Acar and Guy E. Blelloch. *Algorithms: Parallel and Sequential*. Pittsburgh, USA: Carnegie Mellon University, Department of Computer Science, Feb. 2019.
- [4] Tosiron Adegbija et al. “Microprocessor Optimizations for the Internet of Things: A Survey”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* PP (June 2017), pp. 1–1. DOI: <http://dx.doi.org/10.1109/TCAD.2017.2717782>.
- [5] G.M. Amdahl. *Validity of the single processor approach to achieving large scale computing capabilities*. In: *Proceedings of Spring Joint Computer Conference*. New York: ACM, 1967, pp. 483–485.
- [6] Aydin Buluç and John Gilbert. “Highly Parallel Sparse Matrix-Matrix Multiplication”. In: *CoRR* abs/1006.2183 (Jan. 2010).
- [7] Jeffrey R. Chasnov. *Introduction to Numerical Methods*. The Hong Kong University of Science and Technology, 2012.
- [8] Frank Devai. “The Refutation of Amdahl’s Law and Its Variants”. In: Sept. 2018, pp. 79–96. ISBN: 978-3-662-58038-7. DOI: http://dx.doi.org/10.1007/978-3-662-58039-4_5.
- [9] Gabriel Edgar. *An Introduction to Parallel Computing*. URL: http://www2.cs.uh.edu/~gabriel/courses/mpicourse_03_06/Introduction.pdf.
- [10] David Emerson. *Introduction to Parallel Computing Using Advanced Architectures and Algorithms*. Sept. 1997, pp. 1–40. URL: <https://www.researchgate.net/publication/321151707>.
- [11] Prof. Robert van Engelen. *Parallel Programming Models*. URL: <https://www.cs.fsu.edu/~engelen/courses/HPC/Models.pdf>.
- [12] Daniels220 at English Wikipedia. *SVG Graph illustrating Amdahl’s law*. [Online; accessed October 29, 2019]. Apr. 2008. URL: <https://commons.wikimedia.org/w/index.php?curid=6678551>.

- [13] Pawel Gepner and Michal Kowalik. “Multi-Core Processors: New Way to Achieve High System Performance.” In: Jan. 2006, pp. 9–13. DOI: <http://dx.doi.org/10.1109/PARELEC.2006.54>.
- [14] J. Holt et al. “Software Standards for the Multicore Era”. In: *IEEE Micro* 29.3 (May 2009), pp. 40–51. DOI: <http://dx.doi.org/10.1109/MM.2009.48>.
- [15] Jovan Ivković and Jelena Ivković. “Analysis of the performance of the new generation of 32-bit Microcontrollers for IoT and Big Data Application”. In: Mar. 2017.
- [16] George Karniadakis and Robert Kirby. “Parallel Scientific Computing in C++ and MPI : A Seamless Approach to Parallel Algorithms and their Implementation”. In: (Jan. 2003). DOI: 10.1017/CB09780511812583.
- [17] Asanović Krste et al. “The Landscape of Parallel Computing Research: A View from Berkeley”. In: (2006). URL: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html>.
- [18] Vipin Kumar et al. *Introduction to parallel computing. Design and analysis of algorithms*. Vol. 2. Jan. 1994.
- [19] Nigel Lesmoir-Gordon. “THE MANDELBROT SET, FRACTAL GEOMETRY AND BENOIT MANDELBROT – The Life and Work of a Maverick Mathematician”. In: *Medicographia* 34 (June 2012), p. 353. URL: <https://www.researchgate.net/publication/270285889>.
- [20] Sheik Dawood M. “Review on Applications of Internet of Things (IoT)”. In: (Dec. 2018). URL: <https://www.researchgate.net/publication/329672903>.
- [21] Ami Marowka. “Analytic Comparison of Two Advanced C Language-Based Parallel Programming Models”. In: *Parallel Research Lab* (2004). DOI: <http://dx.doi.org/10.1109/ISPDC.2004.11>.
- [22] Ami Marowka. “Think Parallel: Teaching Parallel Programming Today”. In: *IEEE* (2008). DOI: <http://dx.doi.org/10.1109/MDS0.2008.24>.
- [23] Tim Mattson, Beverly Sanders, and Berna Massingill. “Patterns for Parallel Programming”. In: (Sept. 2004).
- [24] Zhang N. “A Novel Parallel Scan for Multicore Processors and Its Application in Sparse Matrix-Vector Multiplication”. In: *IEEE Transactions on Parallel and Distributed Systems* 23.3 (Mar. 2012), pp. 397–404. DOI: <http://dx.doi.org/10.1109/TPDS.2011.174>.
- [25] John Owens and U.C. Davis. *Embracing Industry 4.0 and Rediscovering Growth*. 2019. URL: <https://www.bcg.com/capabilities/operations/embracing-industry-4.0-rediscovering-growth.aspx>.
- [26] John Owens and U.C. Davis. *Parallel Programming Models Overview*. 2008. URL: <http://s08.idav.ucdavis.edu/owens-parallel-prog-models-overview.pdf>.
- [27] Kota Sujatha et al. “Multicore Parallel Processing Concepts for Effective Sorting and Searching”. In: *IEEE* (2015). DOI: <http://dx.doi.org/10.1109/SPACES.2015.7058238>.

- [28] Karlsruhe Institute of Technology. “Multi-core processors for mobility and industry 4.0”. In: *PHYSORG* (2016). URL: <https://phys.org/news/2016-12-multi-core-processors-mobility-industry.html>.
- [29] Klaus-Dieter Thoben, Stefan Wiesner, and Thorsten Wuest. “”Industrie 4.0” and Smart Manufacturing – A Review of Research Issues and Application Examples”. In: *International Journal of Automation Technology* 11 (Jan. 2017), pp. 4–19. DOI: <http://dx.doi.org/10.20965/ijat.2017.p0004>.
- [30] Gudula Rünger Thomas Rauber. *Parallel Programming for Multicore and Cluster Systems*. Germany: Second Edition, Springer Verlag, 2013.
- [31] Massimo Torquati et al. *Smart Multicore Embedded Systems*. New York, Heidelberg, Dordrecht and London: Springer Verlag, 2014. DOI: <http://dx.doi.org/10.1007/978-1-4614-8800-2>.
- [32] Balaji Venu. “Multi-core processors - An overview”. In: (Oct. 2011). URL: <https://www.researchgate.net/publication/51945986>.
- [33] Dragan Vuksanović, Jelena Vešić, and Davor Korčok. “Industry 4.0: the Future Concepts and New Visions of Factory of the Future Development”. In: Jan. 2016, pp. 293–298. DOI: <http://dx.doi.org/10.15308/Sinteza-2016-293-298>.
- [34] Zhu Yang-Ming and Cochoff Steven M. “Medical Image Viewing on Multicore Platforms Using Parallel Computing Patterns”. In: *IEEE* (2010). DOI: <http://dx.doi.org/10.1109/MITP.2010.62>.
- [35] Nan Zhang. “Working towards efficient parallel computing of integral images on multi-core processors”. In: vol. 2. May 2010, pp. V2–30. DOI: <http://dx.doi.org/10.1109/ICCET.2010.5485338>.
- [36] Yousaf Zikria et al. “Internet of Things (IoT) Operating Systems Management: Opportunities, Challenges, and Solution”. In: *Sensors* 8 (Apr. 2019), pp. 1–10. DOI: <http://dx.doi.org/10.3390/s19081793>.