mynatix

# Whitepaper

## LACOS
AI Based
Code Tailoring

# Abstract

**Latency-Optimized Code Segmentation (LACOS)** is the core of an expert-system built by Mynatix, enabling any programmer to exploit the performance of modern platforms without prior knowledge of processor design. This enables software code to optimally benefit from the available performance of a hardware at significantly lower cost than with manual optimization, even for applications where no frameworks exist.

Mynatix is introducing this novel, proprietary method for segmenting a complex (parallelizable) process with unlimited interdependencies into its smallest granularities suitable for automatic optimization.

This technology is a breakthrough for today's data-driven world, where the demand for faster and more efficient computing solutions becomes more and more pressing.

# Content

# Introduction

A common issue in many fields is to identify the best way to divide work between workers in order to get things done in the fastest and most efficient way. To do this well, it is important to determine how the tasks are connected to each other and which tasks can be done separately. Think about building a house: mounting the windows is only possible after the brick-work is finished while putting in the wiring needs different skills to carpentry work.

Running software on modern computers presents similar challenges to those confronting those working in teams. The better a software code can be distributed among different processing cores, the faster (time savings) or more efficiently (energy savings) the computation can be done. The software development market is growing rapidly and architectures providing high, parallel computational power are ubiquitous. The optimal use of hardware is key to keep pace with the exponential growth of data processing and the complexity of modern computational challenges. To meet this need, a software code ideally must be tailored to the hardware on which it will be run.

Even more there is a high distance in today's coding-stacks from the hardware, meaning the gap between programming software and its ability to influence the execution on a specific hardware is getting bigger and bigger[ 1 ]. This effect is particularly visible in the popularity of easier to use programming languages but also in the shift to cloud computing instances, where the runtime-environment is more abstracted from the hardware, than in on-premise solutions. Anyhow, the adaptation, or even customization, of software to hardware is essential for the speed and the efficiency of computation on a modern device. Frameworks[1] and sets of defined functions exist for different architectures of modern devices. They help programmers to adapt their codes and make optimal use of the architectural characteristics. However, it needs a lot of knowledge, experience and especially labor time to handle these support packages efficiently and optimally in a code.

**Welcome to a new age of assisted software development!**

LACOS, an AI-core technology mynatix has developed, enables programmers to have access to an expert system applying code optimizations to exploit parallel and distributed computing capabilities of their software. How fast or efficient a code is running on a device depends on many factors. One is on how a code is going to be adapted to a specific hardware. This adaptation starts by designing an algorithm, followed by writing an appropriate code in a suitable language and utilizing frameworks and libraries to use the available hardware correctly. Therefore, performance and optimization considerations (how to optimally fit code to a target device) are important topics in different phases of the software development life cycle (SDLC). This is independent of the current software development philosophies. In contrast, the parallel computing capabilities of a hardware are physically given. LACOS from mynatix operates at this interface between code and hardware.

[1] A framework is like a construction kit for software developers. Instead of starting from scratch every time they want to build a new app or website, they use a framework that already provides basic tools and rules. It's like a pre-built fundament that they can build their project on without having to build everything themselves. This saves time, keeps things tidy, and helps create better software.

Using LACOS enables this new expert system to operate based on physical laws, and[2]

a. retrieves the parallel opportunities in a code by generating parallel code segments from code statements/instructions;[3]

b. can store these segments in a generic form, which makes it possible to obtain code suitable for different parallel architectures and communication models in one go, i.e. shared memory and message passing;

c. opens new possibilities of automatically optimizing the parallelization/distribution of computation based on static code analysis.

This enables the tailoring of code using frameworks in a novel, generic and automatic form applying the same approach as highly skilled experts. It allows programmers – independent of their experience or specific knowledge – to use and/or switch between the benefits of devices supporting parallel computing differently. This reduces time and complexity in software development in many fields and for a wide range of languages and applications. Code exploiting the parallel computational power of a hardware increases the speed of software and improves the efficiency during the execution. Additionally, LACOS can make parallel/distributed computing worthwhile for new application fields – as it's works automatically and therefore free of any labor time.

[2] achieved by extending the known data-analysis in compilers by including the information on parallelism hidden in Read-After-Read accesses.
[3] as a function of runtime-variables. Runtime-variables are information only known when the program runs.

# Why LACOS

Modern devices and cloud computing services provide the possibility to run software on more than one core and/or even on special units suitable for parallel computing, such as GPUs. Thus, an important topic in optimization of code performance is how to exploit opportunities for parallel execution on more than one core [2,3] and/or a specific computing unit (e.g. GPU). Such performance considerations convolute the software design step, introduce complexity during the implementation step and can make the deployment step more challenging. Besides the inherent code properties and the design of the hardware, the programmers' skills, knowledge and experience influences the possibility to exploit potential performance increase through tailoring the code to the hardware. The skills of developers depend on their work experience and education in certain application fields. Software code that efficiently uses parallel features of hardware runs at higher speeds and therefore creates lower runtime costs. But both implementing and tuning parallel performance can be a very time-consuming and expensive work – especially since it has to be repeated over and over again to adapt to new hardware and code versions during the lifespan of a software application. However, thanks to the LACOS-method, the developer can use this time for other things or simply save it, which significantly reduces the cost of software development.

# Market potential

The software development market in 2023 is forecast to be $251bn and is expected to increase to $1'010bn by 2030, with a CAGR of 21.9% [4]. We estimate that 2% of software development and deployment work is dedicated to general performance issues [6,7,8,9], resulting in a total addressable market of $5-20bn. Not all performance issues are related to parallelism, but with the number of cores per device increasing, it is a growing problem [10]. Therefore, we estimate our serviceable obtainable market of 5% or $ 250mn (rising up to 1bn in 2028 in accordance to the forecasted growth) of the total addressable market.

There is a lot of competition between manufacturers who strive to deliver optimized chips with technically improved (parallel) architectures for different applications [5]. This inevitably leads to rapidly changing architectures. Not every software benefits from parallelism in the same way: AI computations need different properties than office applications. Overall, the growth in data and the demand for more computational power are eminent and steadily increasing (key word "data deluge" gap) [11,12]. After years of stagnation, the latest rankings again show growing interest in languages such as C++ [17]. This relates to its good performance properties and capability to write hardware-near code. But for years, the popularity of interpreted languages such as Python is still unbroken [14]. The number of software engineers is increasing [15] and proves the attractiveness of low-code-programming [16] also for developers with little knowledge of the underlying computing technology. Providing a service that tailors a code to the characteristics of a device by rewriting code-sections with parallel opportunities enables developers to reach new limits quickly and without specific expertise [13].

Mynatix' service uses existing and well-established frameworks but requires no effort for the programmer to add them to the code. Mynatix' proprietary technology can meet the needs of developers tailoring their codes to different devices, including the use of the grow-ing and scalable cloud computing services in a cost-optimized manner. In summary, LACOS-service enables developers to:

- focus on their real skills–developing creative and new features
- reduce time and cost of the whole development cycle
- achieve a shorter time-to-market
- use features from specialized developers/experts
- reduce the dependency on changes in framework and architecture
- improve the speed and/or efficiency of code
- reduce (cloud) runtime costs
- tailor code for applications where optimization was not worthwhile so far because of cost or lack of knowledge

# The Technical Background

LACOS tackles a long-standing problem: detecting parallelism in code and adapting the code to automatically exploit the parallel capabilities of a hardware in a generic way. LACOS is based on physical laws, patent protectable and extracts parallel opportunities in serial code that is based on code design or programming language characteristics. LACOS is the core of an expert system[4] supporting programmers through important steps during software devel-opment. Data analysis in code statements is well established in compilers and they deliver excellent performances targeting single cores. LACOS is able to exploit more parallelism than by data analysis in state-of-the-art compilers by extracting the inherent information in Read-After-Read information not made use of in today's dependency analysis. These new capabilities of the enhanced analytics of LACOS were successfully demonstrated for known problems of older methods in Proofs-of-Concept (PoCs) studies underpinned by Innosuisse. The positive PCT-patentability report shows the novelty and applicability of the method in a key step that challenges most codes: from a programming language to a machine code.

LACOS enables the automated rewriting of code by using frameworks and libraries, which exploits which exploits the power of parallel hardware. It can scope with loop sections that are poorly handled by known methods and is able to automatically rewrite these sections to frameworks such as OpenMP, CUDA or MPI. This is a novel step and can be used to adjust and even completely rewrite code for different parallel devices without any interaction of a programmer. LACOS is not only beneficial for code that is written in a serial form. Code can also be serial bounded, because it is programmed with languages that are inherently serial or limited to use parallel hardware. For example, Python is a widely used language, e.g. in data science and other fields, with support of a lot of different packages. The programming language is an interpreted language, but serially locked by design. There exist efficient solutions for speed-up such as numba[5] , but these require proper use by a programmer. Even in the best case, it can speed-up the code to the limits of a compiled language. C++ is such a language and can be optimized with state-of-the-art compilers. But the code cannot be directly parallelized, nor can the code-sections be rewritten to use frameworks as OpenMP or CUDA. Such frameworks must be applied in a correct way, otherwise they can slow down a code.

LACOS, in contrast, can carve out sections with high parallel potential and rewrite these by properly using such frameworks to achieve the performance of parallel computing. Other languages support parallel or concurrent programming by using additional libraries, but it needs experience to use them properly. For example, Java has libraries and functions to exploit concurrent threads. However, developers must be experienced and apply the libraries correctly to use them.

LACOS supports the development of code not only in rewriting code, but also by highlighting parallel opportunities directly in code editors or integrated development environments (IDE). This is extremely useful for developers, as it is challenging to recognize these opportunities during programming. Often parallelism is a function of runtime properties. Beyond that, LACOS covers this dynamically and provides information on how to use suitable parallel frameworks depending on runtime-variables. This enables developers to adapt the code correctly for the use of the software.

[4] "In artificial intelligence, an expert system is a computer system emulating the decision making ability of a human expert. Expert systems are designed to solve complex problems by reasoning through bodies of knowledge (....)" - Wikipedia
[5] https://numba.pydata.org/

In turn, this also influences the deployment of cloud applications. Today's cloud computing instances differ in architecture, costplans and design requirements of the mostly containerized software. Based on the information from LACOS, the best performing, and most economical cloud provider (by the number of needed instances) can be chosen automatically for the code sections that require high-computational power. These are the sections relating to highest costs in terms of computation. LACOS speeds up the development AND reduces runtime-costs on modern cloud infrastructures.

# Summary – innovation by mynatix

The speed of changing architectures with an increasing number of computing units and parallel computing capabilities can be seen in all devices, from multicore-CPUs in home-PCs to the growing cloud computing solutions with different architectures and services. The number of programmers is increasing due to the growing demand for software solutions. With the rise of low-code methods, an increasing amount of programming is done with languages abstracting the characteristics of the target device.

The wide fields of applications in today's software market leads to programmers with very different backgrounds, skills and experience. Frameworks and libraries only enable those who have the skills, experience and time to apply them properly to harvest the parallel computing power. This gives great potential for an expert system to close this gap. mynatix is introducing LACOS, which is the core for an expert system tailoring code to a device. It carves out the finest physical granularity of sequential sequences in a code and stores these segments in a generic form. This generic form allows the segments in a parallel code to be composed depending on the parallel architecture of a device. Code-sections with parallel opportunities can be rewritten automatically to a chosen target framework . As shown in PoCs and underpinned by an Innosuisse pre-study, LACOS detects more parallel opportunities than data analysis in today's compiler solutions and can relate these to the physical properties of parallel hardware. The LACOS based expert system enables programmers to use frameworks and libraries for using modern devices in the same way as experts do and to create solutions only possible with the speed benefits known from parallel and distributed computing.

The use of LACOS reduces implementation time as well as the need for expensive experts with special skills in hardware design. This directly reduces development costs followed by the additional benefit of improved software speed and cost efficiency at runtime. It can also create optimized codes for parallel architecture even for applications where an optimization was not worthwhile up till now.

# References

1. https://dafoster.net/articles/2014/12/20/languages-by-hardware-distance/
2. https://www.hp.com/us-en/shop/tech-takes/parallel-computing-and-its-modern-uses
3. https://www.codingninjas.com/studio/library/parallel-processing
4. https://www.maximizemarketresearch.com/market-report/global-application-development- and-deployment-software-market/57215/
5. https://www.marketwatch.com/press-release/parallel-system-market-a-detailed-examination-of-the-market-overview-and-growth-outlook-and-its-projected-growth-at-a-cagr-of-45-from-2023- to-2030-2023-06-21
6. Woodside, Murray & Franks, Greg & Petriu, Dorina. (2007). The Future of Software Performance Engineering. Engineering (FOSE), IEEE. 171-187. 10.1109/FOSE.2007.32.
7. Vincenzo Ferme and Cesare Pautasso. 2017. Towards Holistic Continuous Software Performance Assessment. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (ICPE '17 Companion). Association for Computing Machin-ery, New York, NY, USA, 159–164. https://doi.org/10.1145/3053600.3053636
8. Mark Grechanik, Qi Luo, Denys Poshyvanyk, and Adam Porter. 2016. Enhancing Rules For Cloud Resource Provisioning Via Learned Software Performance Models. In Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE '16). Association for Computing Machinery, New York, NY, USA, 209–214. https://doi.org/10.1145/2851553.2851568
9. Jatinderkumar R. Saini, & Vikas S. Chomal. (2020). On Effort Distribution in Software Project Development for Academic Domain. Inter-national Journal of Engineering and Advanced Technology (IJEAT), 9(3), 1755–1761. https://doi.org/10.35940/ijeat.C4706.029320
10. https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
11. High Performance and Embedded Architecture and Compilation, HiPEAC Vision 2015 Report
12. Morley, Jessica & Morton, Caroline & Karpathakis, Kassandra & Taddeo, Mariarosaria & Floridi, Luciano. (2021). Towards a Framework for Evaluating the Safety, Acceptability and Efficacy of AI Systems for Health: An Initial Synthesis. SSRN Electronic Journal. 10. 2139/ssrn.3826358.
13. https://byte-project.eu/the-rising-demand-for-distributed-data-processing-engineers/
14. https://cpduk.co.uk/news/the-growing-demand-for-python-developers-in-2023
15. https://www.griddynamics.com/global-team-blog/number-software-developers-world
16. https://kissflow.com/low-code/low-code-stats/
17. https://www.tiobe.com/tiobe-index/

# software development and parallelism

Software development consists of different phases while the steps of the Software Develop-ment Life Cycle (SDLC) can be simplified for:

capturing system and software requirements

analyzing and describing the problem to be solved;

designing an algorithm and programming the code

testing and deploying the software.

Designing and programming includes the steps to derive an algorithm for a specific problem. An algorithm is a way to solve a problem using a computer.The algorithm is written as a code in a specific programming language. This process is accelerated by using already existing software code packages and known designs. They deliver required functionalities and support to make the code executable on a target device or platform. Performance considerations come up at different phases of the SDLC from design over coding up to the deployment steps. Performance considerations are influenced directly by the complexity and size of the prob-lem, the choice of the programming language, the frameworks and libraries to be used, as well as the available hardware.

For example, in High-Performance Computing (HPC), where large data sets are analyzed and computed, special skills are mandatory to understand the computing problem, as well as knowing how to distribute the heavy computational loads to different units. In data science, the availability of supporting packages with implemented models and known solutions is mandatory to tackle problems. In web-development, standards are important as the code must run on many different browsers or applications. In short,each field of application has its own challenges, norms and solutions concerning performance. On modern devices, parallelism or distributed computing are important, with significant and growing relevance for software development.

Programming is instructing a computer to perform several steps one after the other: do a, then b, then c, and so on. The sequential for of programming code is the easiest approach and results in mostly serial code. In other words, it is much easier and therefore more common to write serial code than parallel code. This is due to the fact that the human brain's ability to do things in parallel is limited and it requires a great level of abstraction and intellectual effort to imagine parallel processes directly. If parallel code is needed, programmers mostly need frameworks and libraries to help bring a code from serial form into one that can utilize parallel computing hardware capabilities.

It is important to note that factors influencing software performance are very complex and intertwined and that parallelism is just one factor. The need for an automatic, generic solution is driven by two factors: a) the increase in platforms with multiple cores and different parallel supporting hardware types; and b) the abstraction from hardware of modern and widely used programming language. The elementary binary step of computing is still the same and the limits of how to compute a program efficiently with a (micro-)processor are physically given

Today's software development takes in a wide field of hardware, software, frameworks and libraries. Since programmers differ in their experience, knowledge and interests, they often have highly individual ways to solve a given problem. Solving problems is the most interesting work for developers – and not taking care of optimizing or adapting a code to a device!

Developers code in human-readable programming languages, the so-called high-level languages. In general, this human- readable code is then translated to machine code by a compiler. The machine code is specific to a particular hardware architecture. A compiler delivers an executable program for a target operating system and hardware architecture. Today's compilers solve this very efficiently for a given architecture with focus on one computing core.

In contrast, the so-called interpreted languages have a wider application field. With these languages, human-readable code is transferred in a series of language specific operations. To run the code, a different software interprets these operations depending on both the soft-ware to be run and the hardware on the target device. In this case, the interpreter (but not directly the code) is optimized by a compiler to a specific hardware. Such programming language abstracts the hardware further away from the programmer. Some of these languages enable code developers to use the parallel opportunities of the target hardware with special functions. Likewise, today's growing shift to cloud infrastructure increases the level of abstraction from the hardware. Cloud instances differ in hardware, price, different available libraries, and support. And this is where LACOS comes into play. It optimizes the code itself – automatically – and bridge therefore the gap of abstraction.

# About

Mynatix is a deep tech startup based in Basel (Switzerland) that focuses on compiler technologies for parallel computing. Its «Latency-Optimized Code Segmentation» principle (LACOS; patent application published and country registrations in process) has shown in proof-of-concepts a high potential for optimizing code to multicore hardware. Basically, LACOS includes Read-after-read (RAR) data dependencies between instructions what enables a novel approach to auto-parallelize software codes. Mynatix aims to learn more about the specific added value of LACOS by developing a first service product for software developers. For additional business opportunities such as Kernel, compiler or chip development, Mynatix is seeking for cooperations, licencing agreements but also (partial) selling of IP.


mynatix inc. | picassoplatz 8 | 4052 basel | switzerland

+41 61 560 24 00 info@mynatix.com www.mynatix.com

https://twitter.com/mynatixcom

www.linkedin.com/company/mynatix

**UID:** CHE-324.320.684

**Subsidary:** mynatix engineering gmbh, basel


mynatix