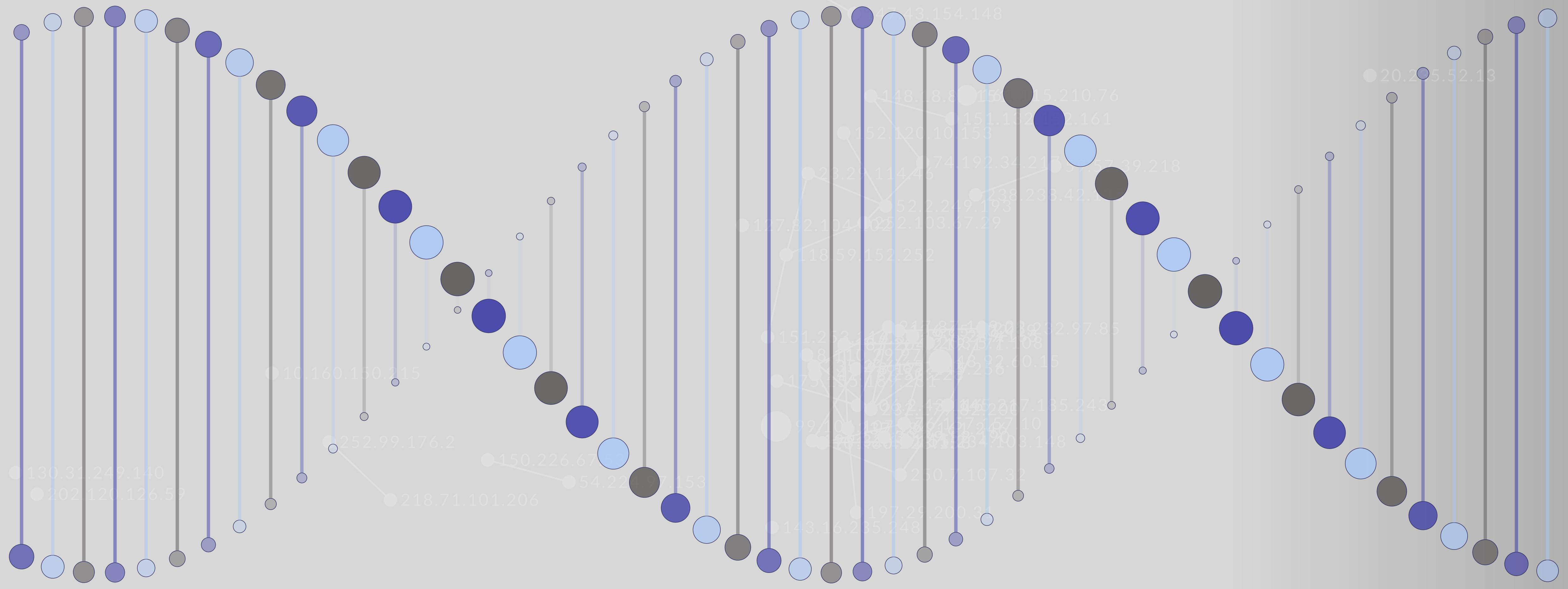
KERNELS
The machine learning system

INTRODUCTION TO KERNELS

EPIDEMIC THEORY
Instead of code distribution

MACHINE LEARNING
Support of the epidemic expansion

STATISTICAL METHODS
Optimize resource usage



Warsaw, PL
+00 1515151515
contact@kernels-analysis.eu

MULTI-AGENT CLUSTER SOFTWARE SYSTEM
Implementation of advanced and highly complex calculations in distributed environments, can be simple, cheap and user-friendly

CONTENTS

I. Introduction

REASONS FOR DEVELOPING KERNELS

AN EASY RIDE TO A CLUSTER PROGRAM

The most important thing should be CONCEPT of the algorithm. Having it, however, often turns out to be only the first step. The developer, after a long cluster setup, gets a collection of instances simultaneously executing across distributed machines. The actual code is translation of something as abstract as the idea into script code for each of them and implementing procedures for how they are to communicate during computation.
KERNELS GIVES READY SUPPORT for virtually ANY parallel algorithm. When you write programs, you only code the logic of the operation, and our system converts it for you to the cluster work.

AN OVERVIEW OF THE ISSUES IN IMPLEMENTING SOLUTIONS LIKE MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

BASIC PROBLEMS of designing learning algorithms for complex data analysis:

- Large scale computing, requires significant resources. This necessitates a distributed infrastructure consisting of many units (CLASTER). It also means the need to ensure their mutual communication and the coding of management processes each time a new program is implemented
- Maintaining stability. The price of computing power implies that failure during any stage of an algorithm run generates significant financial costs and delays.
- Probabilistic nature of the computation, results in the need to dynamically adjust the number of simultaneously executed processes to the available resources.

CURRENTLY AVAILABLE SOLUTIONS performing parallel algorithms on clusters:

- Off-the-shelf implementations that typically provide support for one specific type of parallel algorithm. You put the input into the box and get the result.
- Software that integrates a fixed number of machines into a centrally managed cluster and toolkits that allow you to write communication and management procedures by yourself. The actual computation code requires labor-intensive software under it for these procedures.

BUILDING YOUR OWN CUSTOM SOLUTION in standard clusters:

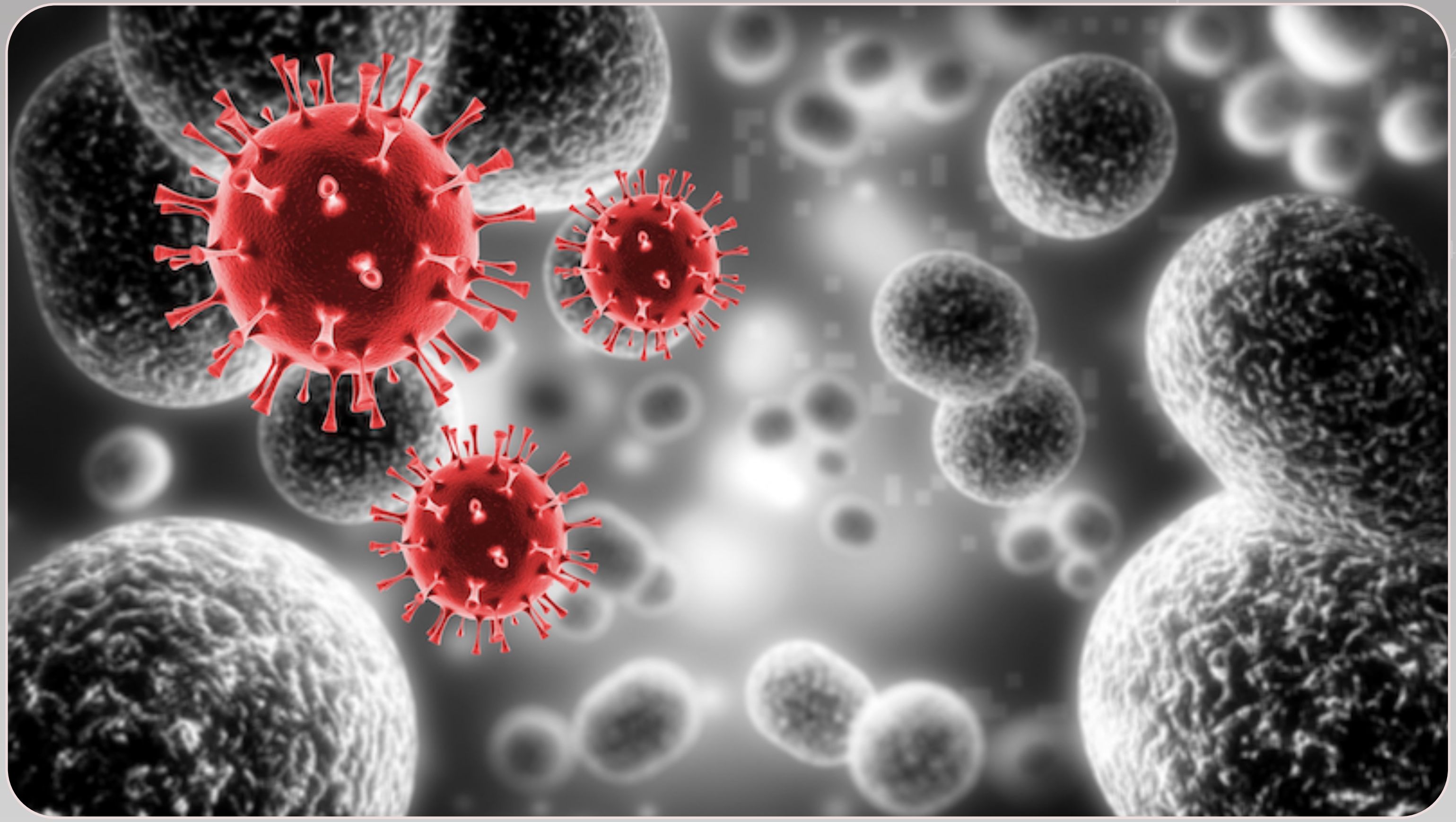
- Develop the structure of its course (CONCEPT).
- INSTALLATION & INTEGRATION tools available within the cluster software offered on the market.
- IMPLEMENTATION. The need to develop both the logic of the progam itself and the procedures that manage the computation within the clustered resources.

I. Introduction

THE USE OF EPIDEMIC THEORY

VIRUS EXPANSION RATHER THAN MANAGEMENT

By treating the cluster as a dynamically changing population distributed across its disjoint units, and USER CODE as VIRUS, KERNELS no longer needs central management. This applied MACHINE LEARNING developed by us, supporting the local decisions of the members of this collective, guarantees the effective and rapid epidemic expansion (computation).
A derivative of the approach is the natural decentralized operation of any parallel algorithm. It carries out DISTRIBUTION OF CALCULATIONS across the cluster instead the developer. KERNELS also provides STABILITY OF WORK by automatically detecting faulty instances and transferring work to other running ones.



EPIDEMIC PROCESS IN TERMS OF DISTRIBUTED COMPUTING

- GENETIC MATERIAL (nucleic acid) - an organic chemical compound, a biopolymer composed of nucleotides. Two basic types of natural nucleic acids are known: deoxyribonucleic acids (DNA) and ribonucleic acid (RNA). It stores a set of instructions for cells called genetic information.

- GENETIC INFORMATION - Information contained in the DNA sequence or, in the case of some viruses RNA, which is passed on to descendant organisms. It can be thought of as a set of coded instructions specifying what the cells of an organism are to do.

- VIRUS - One of the most primordial evolutionary structures. It consists of a coat of protein capsid inside of which is contained nucleic acid (instructions for the cell).

- REPLICATION - The capsid proteins serve as an intermediary to attach to the appropriate receptor on the surface of their host cell (the virus host), allowing them to get inside the cell. Then, with the use of the cell's machinery, they force the cell to make their copies according to the instructions contained in the nucleic acid (the command to build a copy of the virus).

- EPIDEMIC - The process of spreading the virus among its potential carriers. Once the virus enters another host, it forces its healthy cells to reproduce copies of itself, which are passed on, to the next members of the population, during their interactions.

- THE CODE, LIKE A VIRUS, is distributed by infecting a member of the modeled population. Instead of building a copy of itself, it requests, performing the user's calculation. The parallel code execution command available to the programmer means the possibility of task infections from the KERNELS instance performing this instruction. This is done during machine learning assisted contacts of potential carriers. (For details, see section *How KERNELS works?*)

II. Why KERNELS?
BENEFITS FOR YOUR TEAM

ECONOMY OF TIME AND RESOURCES

We provide the product that effectively ACCELERATES PROCEDURES of implementing complex parallel computations, whose scale requires the use of a cluster environment.
It allows the programmer to focus on the most important thing, which is designing the logic of algorithm operations themselves. All the work related to MANAGEMENT OF RESOURCES AND COMMUNICATION between them during the computation WILL BE DONE for the developer by KERNELS..

WHAT WILL KERNELS BRING TO YOUR BUSINESS?

- THE COMPUTATIONS ONLY REQUIRE WRITING THE LOGIC OF THEIR OPERATION. Additional and time-consuming coding of communication within the cluster or control of resources under computation, is not longer needed. The administrative work will be done by KERNELS. This significantly shortens the solution startup time and allows to achieve the desired effect involving a smaller team than before.

- EASE OF CODING NEW ALGORITHMS. Procedures for code parallelization, data sharing, as well as child return reaction are designed to be as intuitive as possible. This allows solutions to be implemented not only by senior development staff.

- OPPORTUNITY TO IMPLEMENT PREVIOUSLY UNAVAILABLE SOLUTIONS. Complex data analysis requires powerful computing resources, forcing the use of distributed hardware infrastructure. The vast majority of solutions available on the market are limited to specific classes of problems only. In the case of KERNELS, it's different. Our system supports virtually any calculation that uses task parallelization.

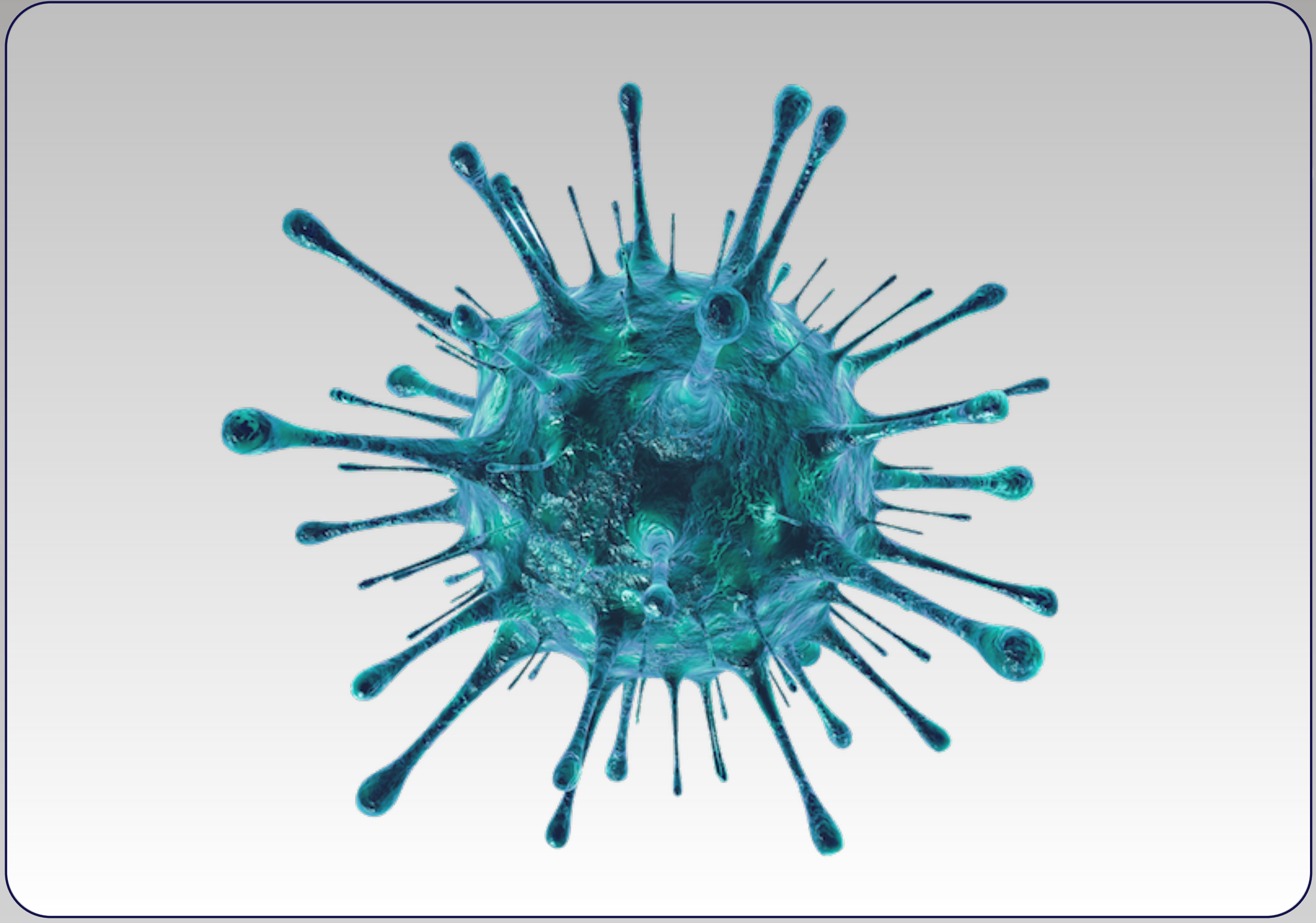
- OPTIMIZATION OF THE HARDWARE RESOURCES USE. Currently, it is expensive to lease or maintain them. Therefore, it's so important that the server is used as much as possible during computations. Idle infrastructure generates unnecessary costs. We have equipped KERNELS with designed by us, machine learning-based software. It optimizes administrative procedures and makes the user's algorithms use the power of the cluster to the maximum extent. (For details, see section *How KERNELS works?,* LOCAL DECISIONS SUPPORT)

- VISUALIZATIONS. Usually the final recipient and verifier of work progress, is a human being. In order to make rational decisions, he needs graphic interpretations more than raw data. KERNELS has a huge number of graphs allowing to present in the interface both final and current data from the work of algorithms.

II. Why KERNELS?
EPIDEMIC THEORY, AND YOUR CALCULATIONS

MORE, FASTER AND STABLE

Using our epidemics concept allows for DECENTRALIZATION not only cluster management, but also the user's algorithm. This solves key problems of distributed programming, such as queuing. It also enables DYNAMIC ATTACHMENT of hardware resources to the already executing code. First of all, it guarantees STABILITY of your work.
Most importantly, the use of KERNELS WILL EXPAND THE SPECTRUM OF possible calculations to include those that were previously unrealistic due to the implementation time.



ADVANTAGES OF USING MACHINE LEARNING-ASSISTED EPIDEMICS IN DISTRIBUTED PROGRAMMING

- NO CENTRAL CONTROL UNIT both managing the computing itself and the entire server infra-structure that implements it. This significantly reduces the risk of queuing, which is a typical problem in the client-server model. It also allows the processing of much larger and more efficient algorithms than was possible in the aforementioned classical model.

- VERY FAST START OF EXECUTION OF ISSUED TASKS, despite the absence of a central management unit. This effect is achieved by replacing client-server management with statistical machine learning methods supporting the expansion of the program code on cluster machines based on the epidemic theory

- STABILITY AND RELIABILITY OF YOUR CALCULATIONS. The failure of a single component of the machine only causes the failure of the operations that are processed on it, not the stoppage of the entire system running the computation. What's more, KERNELS is able to locally detect an interruption and reassign the work in progress to properly functioning cluster resources. Such a problem will not stop your algorithm from executing.

- UNLIMITED SCALABILITY AND SIMPLE PRODUCTION DEPLOYMENT. KERNELS allows dynamic allocation of additional computing power to ongoing tasks without interrupting their processing. The process of adding new resources itself is quick and easy. Just like a supercomputer, but without the steep learning curve of using it.

- SUPPORT FOR UNIVERSAL PARALLEL COMPUTING MODEL. From an implementation point of view, simultaneously executing instances of code can invoke further parallelization, which allows the construction of algorithms that have a tree structure (For details, see section *Universal Model*).

II. Why KERNELS?
SECURITY OF THE SOLUTION

PROTECTION OF EVERY ACTIVITY IN KERNELS

The technology development makes life easier and opens previously unavailable opportunities. However, along with it comes great risks. One of them is gaining unauthorized access to the tools, resources and results of your work.
Thanks to MOST MODERN CRYPTOGRAPHY STANDARDS, as well as HIGH LEVELS OF SECURITY during COMMUNICATION between components, KERNELS will allow you to process even highly sensitive and confidential data.

KERNELS ICT SECURITY MECHANISMS

- SECURE ENCRYPTED COMMUNICATION. The exchange of information between the system's components uses state-of-the-art cryptography that has been validated by the global expert community. Authentication of system components is based on elliptic curves, e.g. Curve25519. Confidentiality of transmitted data is ensured by encrypting it with a recognized and proven standard, algorithm AES 256.

- PROCESSING SECURITY. The correctness of the security of the calculation execution process is ensured through the use of SELinux policies. In practice, this means that every KERNELS-initiated process has strict access privileges from the outset, which means that even if unauthorized access is gained, the possibility of system damage or penetration is infinitesimal.

- DATA CONFIDENTIALITY AT REST. Information in the system is stored on hard drives encrypted using a secure algorithm AES 256.

- FORGET ORDINARY IDS AND EASILY STEALABLE PASSWORDS for confirming user identity. Customer access to the system uses cryptographic certificates X.509 authenticated with digital signatures based on elliptic curve P-384 and hash functions SHA-384. Unlike user-entered passwords, they cannot be easily intercepted and used.

- HIGHER SECURITY FEATURES AVAILABLE. For customers requiring even higher security, it is possible to store certificates to the system on external dongles. It is virtually impossible for an attacker to extract certificates from them. Logging in to KERNELS then requires physical possession of a dongle and entering an additional password. The security of such keys is confirmed at EAL-6 Common Criteria level..

II. Why KERNELS?
ARCHIECTURE

CHOOSE THE BEST SOLUTION FOR YOU

The unique structure of the dynamic system components and the relationships between them makes KERNELS able to work on a wide range of configurations. Installation as well as the addition of further machines is quick and very easy.
No own servers are needed. Launching is done within the popular PUBLIC CLOUDS. However, deployment and subsequent expansion can also be done on a minimum-compliant private virtualization infrastructure (ONPREMISE).

INFRASTRUCTURE THAT ALLOWS KERNELS TO OPERATE

BASIC COMPONENTS on which KERNELS is built:

- MACHINE (virtual or physical). - A single stand-alone computing unit responsible for conducting operations, as well as communicating with its counterparts. It has its own operating system and access to allocated resources.
- VIRTUALIZATION - Layer of abstraction between the hardware platform and the operating system. It is used for optimization, allowing sharing of ONE hardware resource by multiple machines (then called VIRTUAL).
- CLUSTER (computational) - a group of interconnected computer units that work together to provide an integrated working environment.
- PUBLIC CLOUD - A collection of virtual machines running on the provider's hardware infrastructure. The customer can access it through dedicated interfaces. As part of its resources, it is possible to launch a CLUSTER.
- INTER-INSTANCE NETWORK - Information exchange layer within a dynamic computing cluster. It takes the form of a dedicated subnetwork between system components.

WHICH SOLUTION TO CHOOSE? It is suggested to pay attention to two factors:

- FLEXIBILITY - Cloud Advantage - Freedom to modify resources, payment only for actual usage, data security based on the highest industry standards with contractual guarantees.
- CONTROL - Power of On-premise solution (Installation and operation on private servers) - Ownership, but also the need to maintain the entire technology stack.

INSTALLATION REQUIREMENTS - KERNELS requires only simple conditions:

- VIRTUALIZATION ENVIRONMENT - Availability of a standardized environment for running virtual machines (On-premise or Cloud).
- ARCHITECTURE - Standard x86_64 compatible hardware platform.

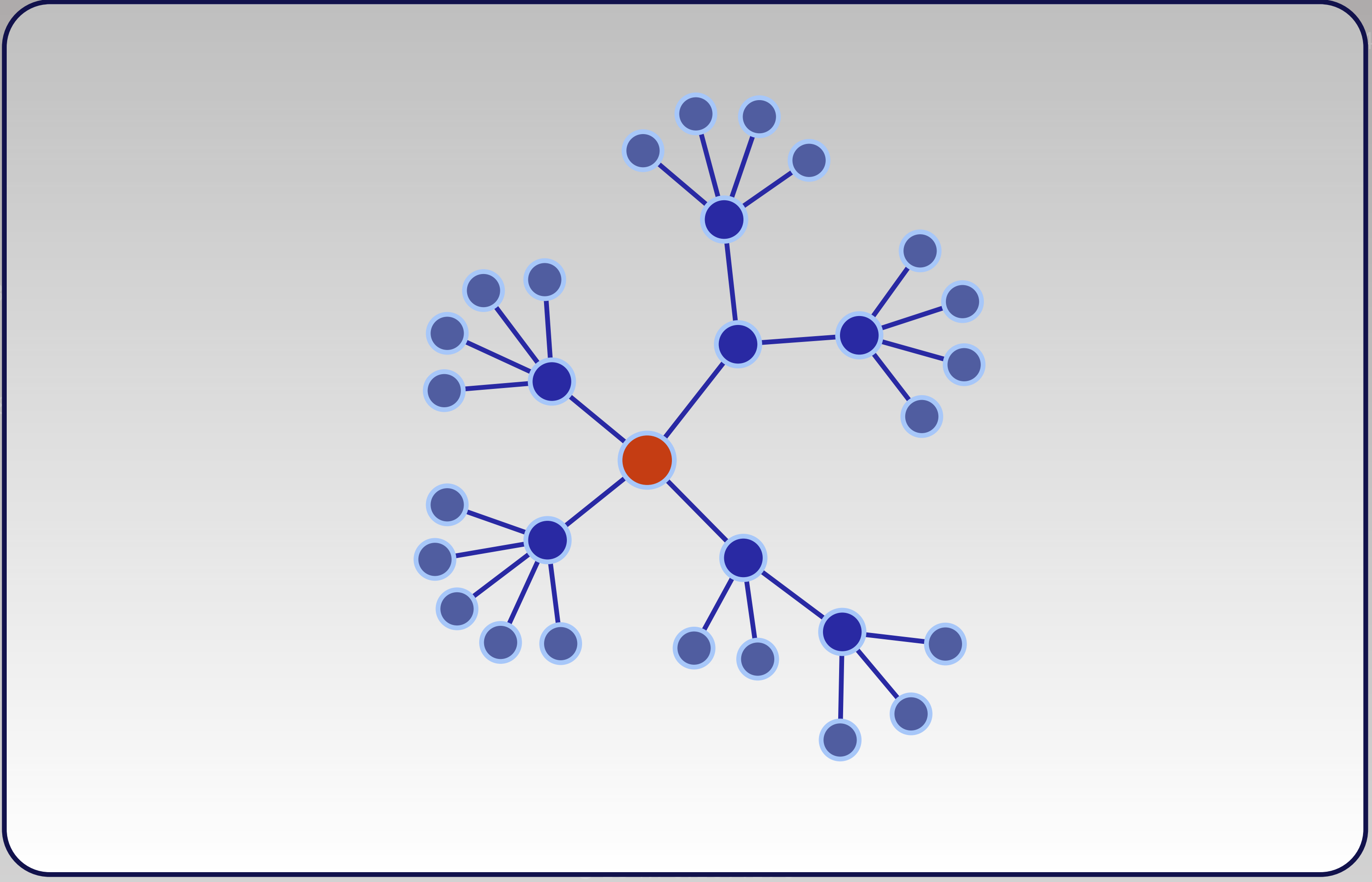
III. What makes us different?

UNIVERSAL CALCULATION MODEL

A MODEL THAT CAN DO ALMOST ANYTHING

Cluster computing does not end with dividing an issue into subproblems and merging the result. Often there is a requirement that each of the simultaneously executing code instances should also be able to parallelize its work. Such algorithms have the structure of a dynamically changing tree. KERNELS will allow you to easily realize ANY ALGORITHM of this kind.

Transition state of the calculation. Shows nodes waiting for results to be returned.



DYNAMIC COMPUTATION TREE AND ITS ADVANTAGES

ASSUMPTIONS OF THE MODEL

The basic element is a node, that is an instance of code that can invoke further nodes and react to the return of their results. The node during its code execution may

- Request execution of any number of child nodes,
- Share any data with all its children.

The parent, that is the node that invoked the children. At the moments of any child result return it can:

- Stop waiting for the other child result returns and move on,
- Request execution of new children and go into wait-for-return mode,
- Go into waiting mode for the return of more children,
- If it was the last child result return, and there was no decision on what to do next, the system recognizes that the parent has completed the work and moves on.

End of the entire calculation is equivalent to completion of the code of the first parent.

WHY IS THIS MODEL SO IMPORTANT?

- It is a response to current issues related to the technology development trends. The huge size of the data from which conclusions must be drawn makes it impossible to analyze all the knowledge at hand. Therefore, solutions are moving toward intelligent sampling. The dynamic computation tree is based on recursive calling of successive nodes. This enables more efficient data mining, by making the number of simultaneous executing tasks (e.g., mentioned sampling) dependent on the effects that are obtained in a given area of the researched resources.

- In other models, such as neural networks, only the end result of the learning can be seen. In the KERNELS, the very structure of its running algorithm shows how it arrived at the final conclusions one by one.

- It is complete. This means that it implements any model of computation that uses parallelization, and thus can implement virtually anything.

III. What makes us different?

KERNELS IN THE MARKET FOR AVAILABLE SOLUTIONS

CLUSTER COMPUTING CAN BE SIMPLE AND EFFICIENT

Distributed programming doesn't have to be different from standard algorithm coding on a single machine. Knowledge of how to operate a cluster and writing a communication layer for each new algorithm, are no longer an obstacle. FROM A DEVELOPER'S VIEWPOINT, KERNELS IS ONE DEVICE. All you have to do, is without touching the server part, invoke nodes and determine the child result return reactions.

KERNELS	Alternative solutions
A collection of decentralized processes traveling around machines, instead of management. Their movement and properties unload queues typical in distributed computing, eliminate the the necessity of maintaining a computation tree, distribute user code across the cluster like a virus, and provide protection against its failure.	After configuration, you get machines (Nodes) and a Scheduler that can initiate a predetermined number of parallel instances on each machine. The implementation of communication between them, and the response to it, is the right computation. However, protection and optimizations are about maintaining the cluster structure, NOT user code.
The implementation of the computing program comes down to nodes programming. KERNELS supports a request execution of any number of children, passing the result to the parent, receiving the return of the child, followed by the option to terminate the entire node including sub-branches.	It is required to code a low-level software, capable of executing simultaneously, static number of instances. There is NO support for nodes, that is, the entire abstract structure of maintaining a dynamic computation tree, request execution of child, response to child return and the node end.
The number of jobs executed in parallel is dynamically adjusted according to the free computing resources on a given machine, rather than predetermined. As a result, KERNELS is able to use the potential of the cluster more efficiently, thereby reducing the cost of running work on it.	There is an assumption that one nodes core can execute one Scheduler-ordered instance. It doesn't matter that at any given time of user program execution, there are free resources for parallel processing of more subtasks Part of the cluster's power remains potentially idle.
Thanks to decentralization, no new setup is needed under the new computation. Whether KERNELS does nothing or executes user programs, running another comes down to entering its code through the interface.	Before running the computation code, you need to specify a lot of configuration parameters for the operating environment. When changing cluster work, or adding a new task, it is usually necessary to stop everything and to re-configure.
Unique support for job parallelization allows you to dynamically increase the computing power of ANY program already in progress.. When the new machines are attached, they will simply start performing the sub-tasks requested by the programs already running.	Dynamically increasing the computing power of a SPECIFIC program requires planning and implementing procedures to determine how new machines are to join in its execution during writing the program code, which often proves to be a non-trivial issue.

IV. How KERNELS works?

A MULTI-AGENT SYSTEM, SUPPORTED BY MACHINE LEARNING

LOCAL DECISIONS SUPPORT (LDS)

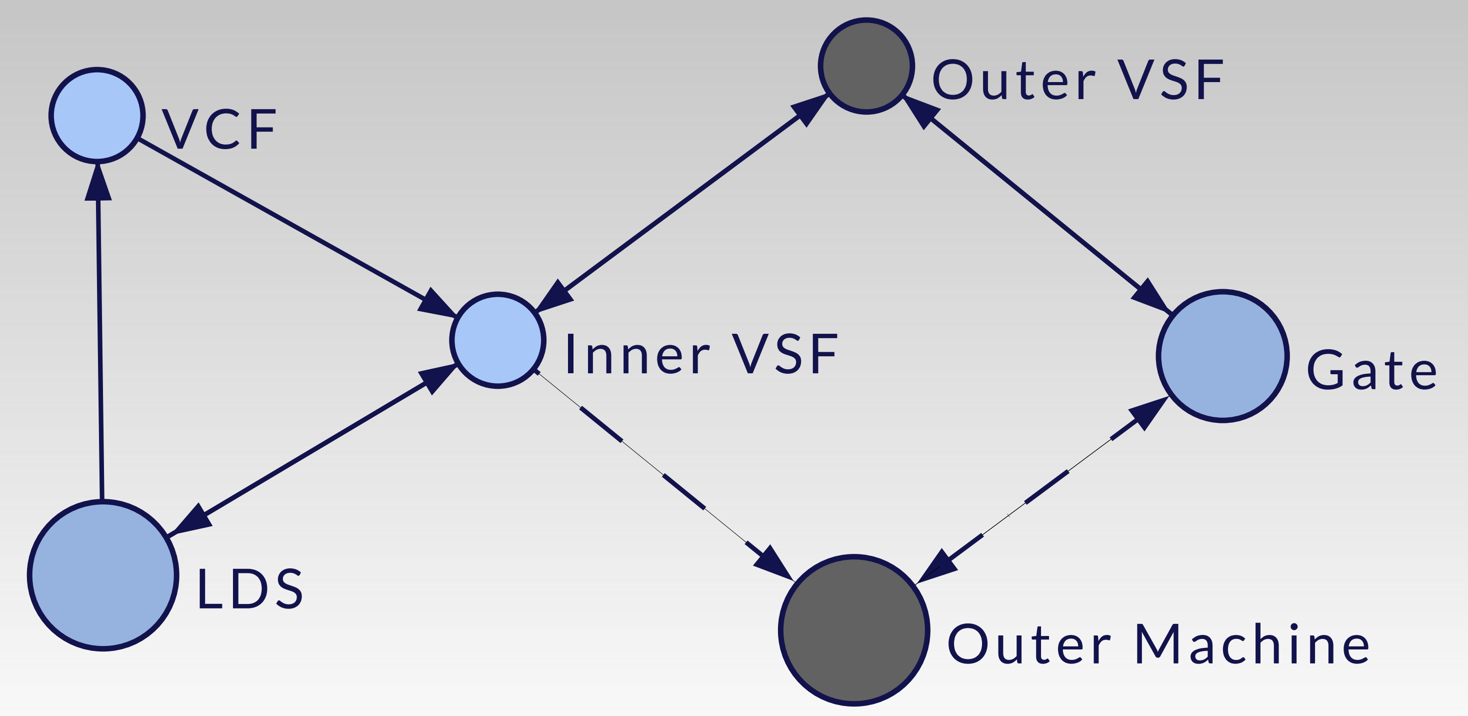
This is our AUTHORIAL SOFTWARE (MACHINE LEARNING). It allows instant distribution of program code within the cluster. Spread across all disconnected machines the LDSs represent a kind of decentralized statistical knowledge of the cluster. Each LDS:

- Learns machine's current operations and recognizes anomalies that occur in its working.
- Based on information provided by VSFs moving throughout the cluster, analyzes the status of external machines.

The purpose of the above activities is:

- Estimating where VSFs can get computational tasks from.
- Dynamic regulation of the VSFs population (code carriers) size and the number of concurrently executed VCFs (computation nodes),
- React to the possible occurrence of machine overload, thanks to RAM and CPU offload procedures designed for this purpose.
- Detect failure of external machines and exclude them from work.

Connections between subprograms of KERNELS



VIRTUAL SEARCH FORK (VSF)

KERNELS IS A MULTI-AGENT CLUSTER SOFTWARE. The role of agents is played by VSF subprograms. Their population spreads the user program (it mean coding of nodes), like a virus. During the VSF life cycle:

- Travels around it initial (INNER) and other (OUTER) machines. Then by communicating with other VSFs, tries to infect itself with a computational task from them.
- Once infected by the task, the VSF makes a request to the LDS to build a computation node and execute user program code by the VCF.
- When a node code implementation contains a child creation command, VSF displays tasks that can infect other VSFs as they interacting with each other.
- Acts as a relay between the User Interface and KERNELS.

VIRTUAL COMPUTATION FORK (VCF)

The program that ultimately executes the node code (written by user). The number of simultaneously executed VCFs changes dynamically based on machine learning and LDS statistical methods. This methodology makes full use of the computational resources of a single machine.

GATE

The program responsible for the transfer of incoming from other machines VSF and handling those that return. In addition, while conducting this work, it collects information about other cluster machines for the LDS.

OUTER MACHINE

Another computing unit that is a copy of the structure shown in the machine diagram: LDS, VSF, VCF, and Gate.

contact@kernels-analysis.eu

+00 1515151515

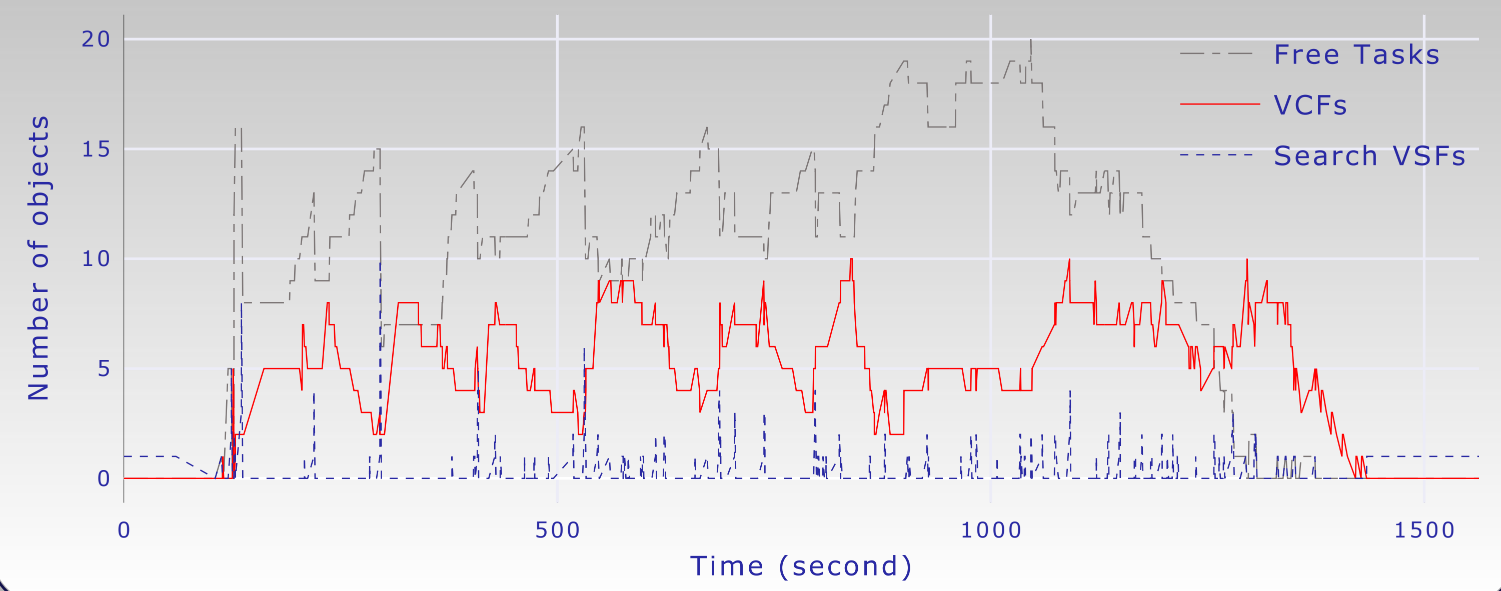
V. KERNELS in action

OPTIMIZATION OF CLUSTER OPERATIONS

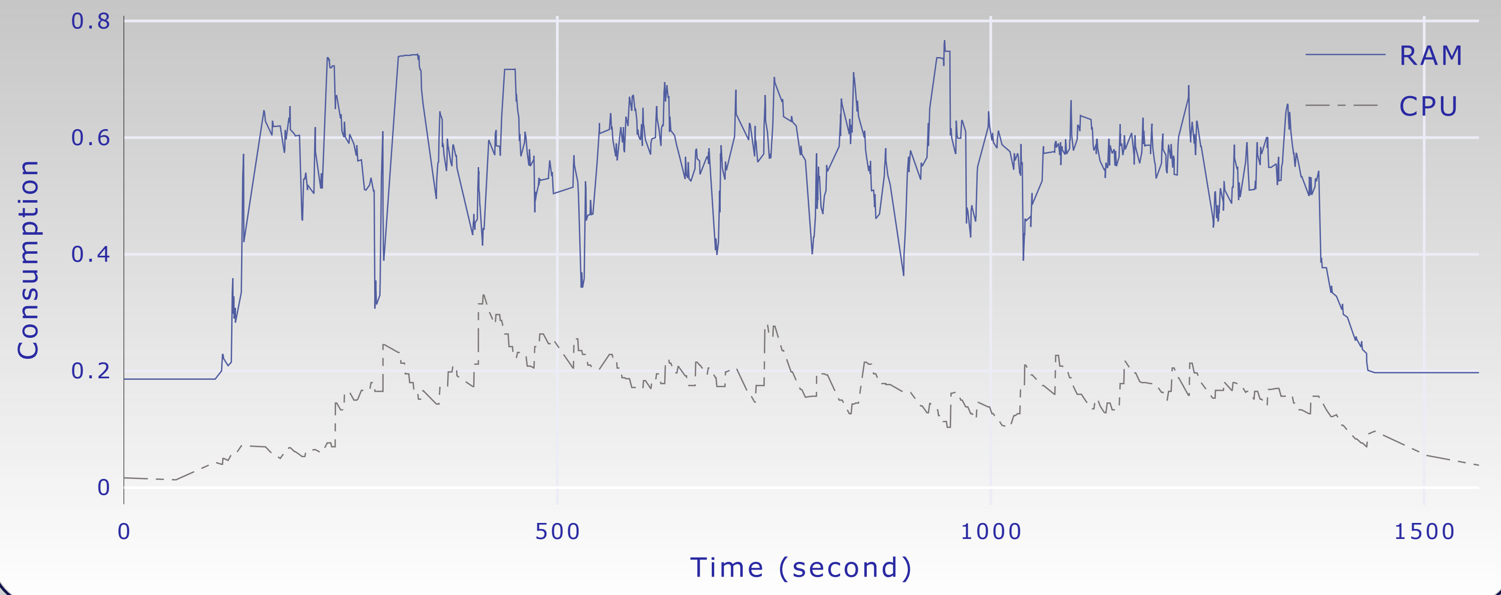
FROM THEORY TO PRACTICE

The example illustrates in charts the advantages of supporting cluster work with MACHINE LEARNING (ML). The demonstration is done on simple code, generating large fluctuations of RAM consumption. Moreover, it explains why this seemingly trivial algorithm, without the use of KERNELS, can become a real challenge for the programmer.

MACHINE LEARNING (FIG. 1)



RAM & CPU CONSUMPTION (FIG. 2)



EFFICIENT AI - HANDLING NON-DETERMINISTIC CODE

ALGORITHM FLOW: The code invokes 5 nodes. Each of them creates randomly between 20 and 30 more. The end nodes occupy RAM in one of the sizes 50MB, 120MB, 140MB, 460MB and keep the memory busy for a random duration of 30 to 60 seconds with probabilities of 0.3, 0.3, 0.3, 0.1, respectively. To put it at risk of overloading, the cluster consists of a single machine with 3 GB of RAM, of which 1.5 GB is occupied by the operating system.

PROBLEMS of non-deterministic algorithms occurring in the example:

- Anomalies that could overload the cluster (sudden drastic RAM increases).
- For most of the program's runtime, the simultaneous execution of tasks requested at a given time (Free Tasks) exceeds the presumed hardware capabilities.
- With that type of code, inefficient management of cluster operation will force computing power overestimation, increasing financial costs unnecessarily.

KRNELS' ML-based decentralized work procedures overcome this handicaps. All thanks to a unique process of adapting the service to the profile of the current work.

MACHINE LEARNING (FIG. 1) adjusts the number of simultaneously processed nodes and the speed of code distribution according to currently free resources.

- Free Tasks - Tasks currently displayed by nodes, awaiting to infect first VSF.
- VCFs - The number of compution work executing in parallel at this point. The ML bases the changes on anomaly-detecting statistical analysis of resource consumption.
- Search VSFs - The number of agents searching for Free Tasks. Due to ML The VSF population adapts to the capabilities of the current parallel nodes execution.

RAM & CPU CONSUMPTION (FIG. 2), despite random memory allocation and outlier occurrences (RAM jumps of 460MB), are held at a safe level.

- RAM (Random-Access Memory) - Operating memory. Exceeding it may cause a task processing failure.
- CPU (Central Processing Unit) - Computing processor. In the example, it has 5 cores. Overloaded it will slow down or practically stop executing commands.

The values of the variables are given after their standardization to the interval [0, 1].

contact@kernels-analysis.eu

+00 1515151515