MISSIONE 4
ISTRUZIONE
RICERCA

SPOKE 1
FUTURE HPC & BIG DATA
FLAGSHIP 5:
Survey of State-of-the-art Codesign - application + SW + HW targeting, benchmarking, patterns, microkernels

Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA | ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

EXECUTIVE SUMMARY

The main goal of flagship 5 is the design, modeling and simulation of heterogeneous accelerators for HPC-cloud systems and edge servers. This goal is planned to be achieved through different objectives. The first objective is related to benchmarking of accelerators, software optimization in heterogeneous architectures, microarchitecture efficiency, performance analysis and portability and profiling of HPC codes. The second objective consists of the selection of mini-applications and benchmarking from multiple domains, such as: AI/ML, Big Data, fluid dynamics, multi-scale simulations, data analysis for astrophysics, N-body dynamics, social media network analysis, graph analytics multi-particle, long-range interacting systems; computational geometry; light-matter interaction, low dimensionality systems, quantum materials, numerical analysis, etc.). Algorithmic prototyping, and algorithmic co-design: simulation, modeling, optimization. Finally, the third objective is related with codesign HW Development and exploitation of next generation of HPC systems: VLSI and FPGA-based architectures; CPUGPU algorithms, VLSI and FPGA-based architectures; data-driven parallelism, data affinity and data locality, streaming computation.

Flagship 5 is carried out by the following units: UNICT(Leader); UNIBO, UNITO, UNIPI, UNIPD, ROMA-TOV, UNINA, PoliMI, UNICAL, INAF, CINECA, ENEA, IIT, UNIFE.

In this report we provide a survey of state-of-the-art approaches and gap analysis of mini-application, benchmarks, and accelerated applications (Section 1). In Section 2 we summarize the cooperation between the different units and with external research groups. A reference list is provided in Section 3.

For subsections with one subindex, namely Section 1.x, we listed at the end of the title the units involved in that specific research. The units are not repeated for subsections involving more than one subindex, such as Section 1.x.y.

Contributors:

Sebastiano Battiato, Sebastiano Boscarino, Armando Coco, Patrizia Daniele, Orazio Muscato, Giovanni Nastasi, Alessandro Ortis, Corrado Santoro, Vittorio Romano, Giovanni Russo, Laura Rosa Maria Scrimali, *Università di Catania* (UNICT)

Giulio Malenza, Marco Aldinucci, Marco Beccuti, Simone Pernice, *Università di Torino* (UNITO)

Fabio Durastante, Luca Gemignani, Beatrice Meini, *Università di Pisa* (UNIPI)

Daniele Bertaccini, Gianfranco Bocchinfuso, Andrea Clementi, Salvatore Filippone, Ugo Locatelli, Francesco Quaglia, Roberto Verzicco, *Università di Tor Vergata* (UNITOV)

Donato D'Ambrosio, William Spataro, *Università di Cagliari* (UNICAL)

Carlo de Falco, Aldo Ghisi, *Politecnico di Milano* (POLIMI)

Giulia Bertaglia, Walter Boscheri, Giacomo Dimarco, Cristiano Guidorzi, Elisa Iacomini, Fabio Schifano, Lorenzo Pareschi, Luca Tomassetti, *Università di Ferrara* (UNIFE)

Alberto Aloisio, Giovanni De Lellis, Guido Russo, *Università di Napoli* (UNINA)

Walter Rocchia, Sergio Decherchi, Cristian Ciracì, Pietro Vidossich, Massimiliano Pontil, *Istituto Italiano di Tecnologia* (IIT)

Eva Sciacca, Nicola Tuccari, Luca Tornatore, Vincenzo Antonuccio, Alessandro Costa, Gian Luigi Granato, Giuseppe Murante, Alberto Vecchiato, Valentina Cesare, Claudio Zanni. *Istituto Nazionale di Astrofisica* (INAF)

Gianfranco Bilardi, Carlo Fantozzi, Carlo Janna, Leonardo Pellegrina, Andrea Alberto Pietracaprina, Geppino Pucci, Michele Scquizzato, Francesco Silvestri, Fabio Vandin, *Università di Padova* (UNIPD)

Section 1. Survey of state-of-the-art approaches and gap analysis of mini-application, benchmarks, and accelerated applications

The state-of-the-art in optimization in FutureHPC and Big Data is rapidly evolving, driven by advances in hardware and software technologies as well as the growing demand for more efficient and effective computing systems.
In FutureHPC, one of the major areas of research in optimization is the development of new architectures and technologies that can handle increasingly complex and demanding workloads. For example, there has been a significant focus on developing new types of processors such as neuromorphic and quantum processors, which can perform specific tasks much faster and more efficiently than traditional processors.
Another area of research in FutureHPC optimization is the development of new software frameworks and algorithms that can improve the performance of existing computing systems. For example, there has been a lot of work done in optimizing parallel computing techniques such as distributed computing and multithreading, as well as developing new algorithms that can take advantage of these techniques to improve performance.
In Big Data, one of the major areas of research in optimization is the development of new data storage and retrieval techniques that can handle large and complex datasets more efficiently. This includes the development of distributed storage systems.
Another area of research in Big Data optimization is the development of new machine learning and artificial intelligence algorithms that can improve the accuracy and efficiency of data processing and analysis. This includes the development of deep learning algorithms, which can learn from large datasets and make accurate predictions or decisions based on that data.
High Performance Computing is of paramount importance also in supply chain management. Supply chain management is a crucial element of any business strategy. It includes the design, planning, execution, control, and monitoring of all supply chain activities, including finding and storing raw materials, work-in-process inventory, and product completion. To achieve optimal performance, companies are investing in HPC resources to optimize the flow of products and services, and to find successful tools to deliver the right products, accelerate time to market, and maintain efficiencies to each stage of the development.

Section 1.1: Astrophysics and Cosmology (INAF, UNITOV)

High Performance Computing (HPC) plays a crucial role in the field of Astrophysics and Cosmology (A&C) by enabling researchers to simulate complex phenomena and analyse vast amounts of observational data. HPC resources allows to run simulations with high resolutions, complex physics, and massive amounts of particles, which are crucial for studying e.g. the evolution of the universe, the formation and evolution of galaxies, or the behaviour of black holes.

The detailed structure and evolution of galaxies, clusters, and other astronomical objects are highly complex and require simulations with high resolutions. With HPC, researchers can perform simulations with billions of particles, allowing them to model intricate astrophysical processes that are otherwise impossible to study. Moreover, astrophysical simulations often require the inclusion of a wide range of physical processes, including gas dynamics, radiation, magnetic fields, and more. HPC resources enable researchers to simulate these complex physics accurately, helping them to better understand how the universe behaves on different scales. Additionally, HPC resources are also essential for analyzing the vast amounts of data generated by astronomical observations. These data sets can be terabytes or even petabytes in size and require advanced computing resources to process and analyze them effectively.

INAF staff involved in Spoke 1 FL5 is developing software and codes specifically for the following astrophysical applications: i) Cosmological and Astrophysical codes including computational fluid dynamics simulations, N-body and smoothed particle hydrodynamics (SPH) simulations on massively parallel computers and GasDynamics; ii) astrophysics pipelines, data analysis and scientific visualization; iii) mini-applications and benchmarking of Least Square Solutions of Big Systems of linearized equations from astrophysics domains (GAIA). The state-of-the-art of the applications is reported in the following sections.

Section 1.1.1: Cosmological and Astrophysical codes

GADGET

General overview

GADGET [5] is a state-of-the-art Astrophysical highly scalable HPC numerical simulation application and it is a reference in the numerical cosmology community. It solves the hydro-gravitational problem for a collisionless fluid.
The scientific problem can be divided in two parts: gravity, a long-range component affecting all of the computational elements of the chosen domain; and hydrodynamics, that is almost local and only affect normal matter (in astrophysics, called "baryonic" matter).
GADGET computes gravitational forces using a TreePM technique. This means that a mean field approximation is used for large scales – called Particle-Mesh, PM – while at smaller scales a Treecode is used.
In the latter case, the computational domain is partitioned using an oct-tree. For nearby regions of the computational domain, all particles interact among themselves; while (in first approximation) only the center of mass of far regions is considered.
Hydrodynamics is solved using a so-called Smoothed Particle Hydrodynamics technique. In this case, one particle represents a fluid element, whose thermodynamical properties such as density, pressure, entropy, are obtained from those of neighbouring particles, smoothed over a given physical scale (smoothing length), using a kernel with suitable characteristics. The smoothing length gives the resolution of the computation; only information of particles within such a scale is needed for the calculation of hydro forces.
GADGET can work both in "physical" and "comoving" coordinates. This means that the code is well suited both for standard numerical computation and for cosmological ones. An example of the former experiment is the evolution of a model galaxy whose initial conditions are devised so as to represent the properties of the Milky Way as it is observed today. The latter kind of computations usually start from an early phase of the Universe evolution, as deduced e.g. by the data we have on the properties of the Cosmic Microwave Background – almost 13 billion years ago – and follow the formation and evolution of structures in a fully cosmological context, as described by our models, reaching the present time.

GADGET also contains a number of so-called “astrophysics modules”, used to compute more processes and properties, needed to follow in details the formation and evolution of cosmic structures. Among those, of paramount importance are: star formation and stellar energy feedback, given by the explosion of massive stars at the end of their lifecycle, known as SuperNovae type II (SNII); cooling and heating of the gas; energy feedback from Active Galactic Nuclei powered by a central SuperMassive Black Hole; evolution of stars and formation of elements during the explosions of Supernovae or other rarer phases of the life of stars. The majority of those processes cannot be directly computed, on the basis of first principles, like gravity and hydrodynamics. The reason is that the dynamical range needed is by far outside the reach of current calculus power, even using the most powerful existing supercomputers. They are thus modelled with "sub-resolution" models, aimed to capturing the effect on resolved scales of the (astro) physical processes happening at unresolved scales. These sub-resolution models are currently widely employed in the astrophysical literature and are needed to produce state-of-the-art theoretical computations.

Technical details

GADGET is written in C and massively parallelized using a hybrid model, MPI+OpenMP. Most of the physics is also ported on GPU with a significant speed-up. Computing tiles (particles) are assigned to MPI task using domain decomposition. In details, the code computes a space-filling Peano curve that touches every particle in the computational domain. A computational weight is assigned to each particle. The curve is divided into M segments, having similar computational weight, and assigned to the N MPI tasks (M=x*N, where x can range from 1 to some tenth and must be even). This scheme achieves a good workload balance at the expense of memory unbalance. The code is written in C, massively parallelized through MPI+OpenMP and presents a good scalability up to ~10^5 cores, although its inherent computation intensity, in its current implementation, may limit its ability to adapt to a multi-million thread scale.

Memory and disk occupancy are obviously dependent on the size of the problem. As for disk storage, Typical state-of-the-art problems requires ~100TB of disk space, cutting-edge problems reach the ~1PB, target exa-scale problems are as large as ~20PB. The memory occupancy, for state-of-the-art runs, is as large as ~100-200TB.

Run-times are of typical order of 50M core-hours over thousands of cores.

PLUTO
General overview
PLUTO (https://visivo.readthedocs.io/) is a freely-distributed software for the numerical solution of mixed hyperbolic/parabolic systems of partial differential equations (conservation laws) targeting high Mach number flows in astrophysical fluid dynamics. The code is designed with a modular and flexible structure whereby different numerical algorithms can be separately combined to solve systems of conservation laws using the finite volume or finite difference approach based on Godunov-type schemes.

Equations are discretized and solved on a structured mesh that can be either static [6] or adaptive [7]. The Adaptive Mesh Refinement (AMR) interface relies on the Chombo library for parallel calculations over block-structured, adaptively refined grids (https://commons.lbl.gov/display/chombo/).

Technical details
The code is written in the C programming language while the AMR interface also requires also C++ and Fortran.

PLUTO is a highly portable software and can run from a single workstation up to several thousands processors using the Message Passing Interface (MPI) to achieve highly scalable parallel performance.

The software is developed at the Dipartimento di Fisica, Torino University in a joint collaboration with INAF, Osservatorio Astronomico di Torino and the SCAI Department of CINECA.

The current release adds Particles support, Hall MHD, forced turbulence and RK-Legendre time stepping for parabolic problems.

FLASH (https://flash.rochester.edu/site/index.shtml)
General overview.
FLASH is an Adaptive Mesh Refinement modular code aiming at numerically modelling the complete MHD system of Navier-Stokes equations, and includes a wide range of source terms accounting for energy losses and sources, including a detailed treatment of thermonuclear energy sources, radiative cooling and cosmic rays transport. It adopts the PARAMESH library (https://opensource.gsfc.nasa.gov/projects/paramesh/index.php) for effective parallel implementation of the adaptive domain decomposition, and a wide selection of refinement/derefinement criteria (up to 13). The hydrodynamic and MHD modules allow the user a rather wide choice of the numerical integration scheme (among others: split/unsplit PPM, WENO, PCM, GP), and the physics modules include (special) relativistic CFD and MHD.
FLASH has been extensively used by a very large, international community of physicists, mostly including plasma physicists and astrophysicists.

Technical details.
The package is written in Fortran90, C, C++ and Python, and parallelized using OpenMP and MPI, and it scales efficiently up to few thousand cores. One of the most critical issues concerning FLASH the I/O. Latham et al (2012) got a significant improvement by adopting Parallel-NetCDF: yet, there is a significant room for further improvement. It would be particularly useful to improve parallel I/O on HDF5 files, as this is one of the most widely adopted formats.

Section 1.1.2: Astrophysics pipelines, Data analysis and scientific visualization

Data analysis and scientific visualisation are crucial for astronomy and astrophysics because they allow us to make sense of the enormous amounts of data generated by modern telescopes, observatories and numerical simulations. The sheer size and complexity of these data sets make it impossible to analyse them manually or with simple statistical tools. Instead, researchers rely on sophisticated data analysis techniques, such as scientific visualisation, visual analytics and machine learning algorithms, to extract meaningful insights from the data. This enables them to uncover new phenomena, identify patterns and trends, and make predictions about the behaviour of celestial objects.

In addition, scientific visualisation is a powerful tool that allows researchers to represent complex data sets visually, making it easier to understand and interpret the data. By visualising the data, researchers can identify patterns and trends that may not be immediately apparent from raw data sets. Moreover, visual representations can help communicate scientific findings to a wider audience, including policymakers, educators, and the general public. For example, scientists can use visualisations to create stunning images of galaxies, stars, and other celestial objects, bringing the wonders of the universe to life and inspiring a new generation of scientists and astronomers.

VisIVO [1] (Visualization Interface for the Virtual Observatory) performs multi-dimensional data analysis and knowledge discovery of a-priori unknown relationships between multi-variate and complex datasets in Astrophysics. VisIVO development started within the activities of the Virtual Observatory framework. It supplies users with functionality to render meaningfully highly-complex, large-scale datasets and create movies of such views using distributed computing infrastructures.

VisIVO (https://visivo.readthedocs.io/) provides an integrated suite of tools and services that can also be used in other scientific fields. The VisIVO suite offers a variety of flavours as follows:

VisIVO Server a platform for high performance visualization.
VisIVO Library for running complex workflows on DCI, clouds and HPC infrastructures to efficiently produce complex views of the dataset and full movies directly with the user-code internal data representation (i.e. without the need to create intermediate files).
VisIVO ViaLactea Visual Analytics (VLVA) [2], developed within the ViaLactea project, which allows to exploit a combination of all new-generation surveys of the Galactic Plane to analyze star forming regions of the Milky Way.
VisIVO ViaLactea Web (VLW), a work-in-progress simplified web version of the VLVA, developed in collaboration with University of Portsmouth (UK) providing an efficient visualisation (GPU and CPU rendering) on remote server.

VisIVO technologies have been demonstrated as success stories in numerous relevant multidisciplinary environments, and Projects. As an open access software, significant interest in its usage has been shown from all over the world (evident by the number of hits on the websites) and several scientists from different domains (e.g. Nuclear Physics) have produced visualisations of their data (simulations or observational datasets).

VisIVO is written in C++ and uses the Visualization Toolkit (VTK) library for the rendering modules. The suite is maintained by INAF-Astrophysical Observatory of Catania and is continuously enriched with several international collaborations.

Section 1.1.3: Mini-applications and benchmarking of Least Square Solutions of Big Systems of linearized equations from astrophysics domains (GAIA)

The Astrometric Verification Unit-Global Sphere Reconstruction (AVU-GSR) Parallel Solver [3] is a code developed for the ESA Gaia mission, whose main purpose is to find, with a [10, 100] μarcsec precision, the astrometric parameters of ~108 stars in the Milky Way, the attitude and the instrumental setting of the Gaia satellite, and the global parameter γ of the parametrized Post-Newtonian formalism. Deriving astrometric parameters (parallaxes, right ascension, declination, and proper motions along those two directions) with such a precision is essential to accurately investigate the formation, evolution, kinematics, and dynamics of our Galaxy. To find these parameters, the code solves a system of linear equations, A x = b, with the LSQR iterative algorithm. The lines of the coefficient matrix A correspond to the stellar observations and the columns of A correspond to the unknowns of the system.
At the end of the Gaia mission, expected in 2025, the dataset is likely to occupy a memory of ~[10, 100] TB, typical for the Big Data analysis, corresponding to ~1011 observations and ~108 stars. With this size, an efficient parallelization scheme is required to obtain scientific results in reasonable timescales and to guarantee an optimal production in perspective of the future Gaia Data Releases.
This code has been in production since 2014 on CINECA infrastructure Marconi100, under an agreement between INAF and CINECA and with the support of ASI. The code was firstly parallelized entirely on the CPU, with a hybrid MPI+OpenMP parallelization scheme, where the computation related to a portion of the lines of the coefficient matrix was assigned to a different MPI process and was further parallelized with OpenMP. To accelerate the code, it was ported to the GPU [4]. In a preliminary analysis, the porting was performed by replacing the high level language OpenMP with the high level language OpenACC, obtaining a modest speedup of ~1.5. The code was successively optimized by replacing OpenACC with the low level language CUDA, to better match the GPU architecture and the topology of the system to solve. This resulted in a substantial acceleration of the code: the speedup of the code over the MPI+OpenMP version increased with the system size and the number of employed GPU resources, reaching a maximum of ~14 for the considered systems.

Section 1.1.4: Computational Methods in Dynamical Astronomy

The application of computational techniques in Celestial Mechanics problems has always been innovative and challenging. In particular, modern space missions without computers would not even have been conceivable.
The problem of the long-term stability of our planetary system is investigated since several decades ago by using computers. This is commonly done by both numerical integrations and algebraic manipulations aiming to construct suitable Hamiltonian normal forms. Since the end of the last century, the interest for this kind of investigations has been enormously increased by the discovery of multiple-planets extrasolar systems. The problem of their stability is made harder by the fact that often the (mutual) inclinations of the exoplanets is totally unknown or (in the best cases) poorly known. Thus, the usual approach based on

(KAM) normal forms here can be fruitfully applied on reverse. Ranges of the possible values of the mutual inclinations are deduced by prescribing the KAM stability of the extrasolar planetary stability, that is ensured by the convergence of the construction of the corresponding normal form (see [8]). Such an approach has been substantially improved by preliminarly constructing the normal form for a so called elliptic torus around which plenty of KAM invariant tori are winding; then, it is easier to construct the final normal form corresponding to the one of them that is characterized by the frequency of motions we are looking for. This combination of normal forms has been successfully applied to the upsilon-Andromedae extrasolar system (that is one of the very few for which also the inclinations are approximately known) and has allowed also to properly define a numerical indicator of the dynamical robustness of an orbital configuration (see [9, 11]).
From a computational point of view, the accomplishment of such a research project has required to fully develop libraries of computer algebra software specially designed to deal with Celestial Mechanics problems and Hamiltonian perturbative methods; this programming effort started more than 20 years ago (see [10]). A question about the generality of our approach naturally arises: can it apply also to other exoplanetary systems? Since for most of them the inclinations of the exoplanets is unknown, the computational complexity increases dramatically: algorithms constructing normal forms must be launched for every point of a regular grid covering the ranges of the orbital parameters given by the observations. Of course, parallelization comes here into play, but each computational procedure of this type requires a very big amount of RAM. Thus, in the near future the success of this investigation about the robustness of extrasolar systems 3D architectures will depend on both a good balancing of all the parameters ruling the computational complexity and the careful implementation of techniques of algebraic manipulations for the saving of memory.

Section 1.2: Applications in life sciences, biomedical, social sciences, material sciences, condensed matter physics (UNIFE, IIT, UNICT, POLIMI)

Commentato [1]: alla sezione 1.2.1 contribuisce anche POLIMI, in collaborazione con IIT

UNIFE focused on the design and construction of efficient algorithms for quantifying uncertainty in complex systems oriented to applications in physics, engineering, biomedical, and social sciences that require multiscale simulations. The development of numerical methods for quantifying uncertainty in complex systems is a rapidly growing area of research that has broad applications in several fields [99-100]. In many cases, simulations of these complex systems require the integration of multiple scales, ranging from microscopic to macroscopic, and involve a large number of uncertain parameters. State-of-the-art algorithms for quantifying uncertainty in these multiscale simulations include stochastic methods, such as Monte Carlo simulations and Markov Chain Monte Carlo methods, as well as data-driven methods, such as machine learning and deep learning techniques [95]. Another class of methods is based on a deterministic approach and makes use of generalized polynomial chaos expansions and stochastic Galerkin or stochastic collocation techniques. These algorithms can be used to propagate uncertainties in model inputs to outputs, providing estimates of the variability and uncertainty in simulation results. Additionally, these algorithms can be used to optimize model parameters, reduce computational cost, and improve model predictions. Overall, the development of efficient algorithms for quantifying uncertainty in complex systems has the potential to improve our understanding of complex phenomena and inform decision-making in a wide range of fields.

IIT staff involved in Spoke 1 FL5 is developing models, software and codes for the applications in life sciences, material sciences, and condensed matter physics. We are considering the following computational topics as particularly relevant and as those which could largely benefit from algorithmic refactoring: i) (bio)electrostatics and electromagnetism; ii) template matching in protein-protein docking and cryo-em; iii) quantum physics; iv) computational geometry.
The state-of-the-art of the aforementioned applications is briefly sketched in the following sections.

Section 1.2.1: (bio)electrostatics, electromagnetism and plasma simulations
Electrostatics and electromagnetism are at the core of countless applications. At IIT, we have a lot of experience in simulating electrostatic effects inherent in biomolecular phenomena as well as electromagnetic effects inherent in light-matter interaction. Many of the underlying models lead to descriptions based on Partial Differential Equations, whose solution can become computationally prohibitive if the size or the complexity of the studied systems are too high. This is where specific solutions, which combine effective models and advanced applied mathematics approaches, can make the difference.
The finite element method (FEM) is a general and versatile numerical method for solving partial differential equations, and it is commonly used in the field of structural analysis, heat transfer, fluid flow, and electromagnetism. Crucial to FEM is the setup and solution of linear systems of equations and represents hence a typical kernel in computational physics.
In this context, sparse direct solvers [12] are very popular since, unlike iterative solvers and preconditioners, sparse direct solvers do not suffer from convergence issues and do not require much tuning. However, efficiently implementing a sparse direct solver in a scalable and high-performance way can be quite challenging. With a good numerical factorization, most of the work in a sparse direct solver is performed using dense linear algebra operations on these dense sub-blocks. Although the numerical factorization phase can achieve relatively high performance on modern multi-core architectures, the fact that many of these blocks are small makes it hard to fully exploit the potential of GPU accelerated nodes. To solve this bottleneck, one would need to minimize data movement between the CPU and the GPU [13], and hence reduce kernel launch overheads, or increase the communication speed between the CPU and the GPU through the development of an ad hoc hardware implementation.
From the point of view of the development of an efficient FEM code, this brings two problems: from the one hand to obtain a good computational performance (considering the memory/compute bound nature of the problem) and on the other hand to assure longevity/portability to the code by avoiding too code-intrusive approaches. In this context, we will develop computational kernels and use libraries that take advantage from recent and future accelerators, and to confer portability and longevity, we will use a OpenACC/OpenMP approach which will allow to support both current and future accelerators. In particular, we will focus on the frequency domain electromagnetic FEM solvers.

Different solutions might be more suitable for applications to bio-electrostatics. In IIT, there is a long tradition of continuum electrostatics models and solvers for biomolecular applications. Specific acceleration of PDE solvers for this kind of applications are done in collaboration with Prof. De Falco at Polytechnic of Milan and are detailed in their contribution.

UNICT is involved in the simulation of a plasma source using the Particle In Cell method, carried out in collaboration with INFN Catania Laboratori Nazionali del Sud (contact person Dr. Lorenzo Neri).
UNICT is dealing with two aspects concerning the plasma simulation: the numerical solution of the Poisson equation for calculating the electric field produced by the charges inside the source, and the choice or study of the most indicated for the advancement of charges in time. The goal of the research is to provide implementations that are computationally time efficient on multicore machines and maintain the necessary level of accuracy. In particular, the new implementation of the resolution of the electrostatic field allows not to use interpolating functions and writes to files necessary with the previous implementation made with the Comsol Multiphysics software. The integration of the equations of motion has also been extended to the relativistic case which concerns a small but not negligible fraction of the electrons involved in the calculation. rest of the simulation code, and a solver for the advancement in time that also takes into account the relativistic corrections, which may not be negligible given the speed of the charges.

Section 1.2.2: template matching tasks in protein-protein docking and cryo-em applications.
The availability of large amount of experimental data calls for advanced and highly computationally efficient methods of analysis and interpretation. We decided to focus here on implementations of the general concept of template matching, where two scalar fields in 3D are compared across many, or all, possible roto-translations. Main computational implementations make use of the FFT, which, however, is only suitable when rotation angles are unbounded. Among the very many possible applications of this routine there is protein-protein docking [14, I-15], very significant in the life science field, and template matching for the recognition of a given macromolecular system in a heterogeneous environment analyzed via Cryo-EM [16, I-17].

Section 1.2.3: quantum physics
Estimating physical observables, as free energies, is of paramount importance both in organic and inorganic chemistry. To reach a high level of accuracy in the estimation process often the Density Function Theory (DFT) level of theory (hence Kohn-Sham equations) is employed yet being this approach significantly time consuming. Additionally, when estimating thermodynamic observables, size effects arise and a fully dynamical (Molecular Dynamics) treatment is needed. This inevitably creates further computational burden as the dynamical simulation of big molecular environments is required. To cope with these issues at EPFL and Sapienza University a code dubbed MaZe (Mass-zero constrained dynamics for simulations based on orbital-free density functional theory) was devised which takes advantage of the fast, albeit approximate, orbital free DFT method. This was coupled with

the recently introduced Mass Zero (from which the MaZe acronym) method which allows to treat the DFT minimization problem as a constraint and hence granting a zero mass, fully adiabatic, decoupling in the spirit of the Car-Parrinello method. Despite the speed of the method, the code is currently not fully optimized to exploit parallel and GPUs architectures.

Section 1.2.4: Computational geometry

The problem of building and analysing molecular surfaces is at the same time challenging and relevant in the computational biophysics field, since it takes part in continuum electrostatics-based implicit solvent models [18], in the estimation of hydration energies, and in the search for potential new binding sites in proteins [19]. We are focusing on the parallelization of some key processes related to the construction of the molecular surface and on its mapping onto a grid, chiefly within the NanoShaper tool which was devised in IIT.

Section 1.3: computational simulation of complex biological systems (UNICT, UNITO, UNITOV)

UNICT work concerns the study of biological network formation models, carried out in collaboration with the King Abdullah University of Science and Technology (KAUST, supervisor Prof. Peter Markowich), and with Prof. Vito Latora (University of Catania and Queen Mary, London).

The mathematical models describe the formation of biological networks such as, for example, the veins of a leaf that carry lymph over the entire surface, or the vascular system of a living being that transports oxygen throughout the body. The starting point of these models is the idea that the network that supplies a certain region is formed by trying to achieve the goal with the least "cost". From the mathematical point of view, this idea translates into a variational formulation, in which the equations that describe the model derive from the minimization of a certain functional that describes the overall cost. The resulting model is described by a system of evolutionary partial differential equations in two or three spatial dimensions. Detailed numerical simulation of such a system presents significant challenges, even in two spatial dimensions, due to the multiscale nature of the solutions. The computer codes that will be developed for the simulation of these problems must therefore be set up in order to scale well on parallel architectures, in order to provide an adequate spatial resolution and capture the solution of the problem at different scales.

The UNITO group is working on the definition of a possible case of study in the context of computational simulation of complex biological systems.

Indeed, computational models are crucial to address critical questions about systems evolution and deciphering system connections. The pivotal feature to making this concept recognisable from the biological and clinical community is the possibility to quickly inspect the whole system bearing in mind the different granularity levels of its components.

This holistic view of system behaviour expands the study of evolution by identifying the heterogeneous behaviours applicable, for example, to the cancer evolution study.

With the recent advances in supercomputers, most of the challenges in modelling and understanding the complexities of biological networks can now be addressed, however, most of the current modelling tools are not able to efficiently scale up on these infrastructures.

In this context our group has recently developed a tool called GreatMOD which represents a new way of facing the modelling analysis, exploiting the high-level graphical formalism, called Petri Net (PN), and its generalizations, which provide a compact, parametric and intuitive graphical description of the system and automatically derivation of the low-level mathematical processes (either deterministic and stochastic) characterizing the system dynamics. The framework strengths can be summarized into four points:
the use of a graphical formalism to simplify the model creation phase by exploiting the GreatSPN GUI;
the implementation of an R package, EPIMOD, providing a friendly interface to access the analysis techniques (from the sensitivity analysis and calibration of the parameters to the model simulation);
a high level of portability and reproducibility granted by the containerization of all analysis techniques implemented in the framework;
a well-defined schema and related infrastructure to allow users to easily integrate their own analysis workflow in the framework.

The architecture of this framework is composed of three main modules which cover different aspects (see Figure 1).

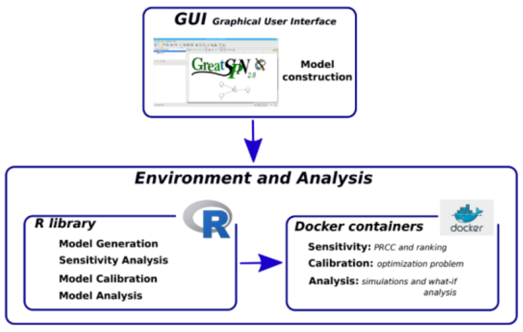


Figure 1: **Framework schema** depicting its modules and its functionalities from a user point of view.

The first module consists of a Java Graphic User Interface (GUI) based on Java Swing Class which allows drawing models using the PN formalism. This graphical editor is part of GreatSPN [20], a software suite for modelling and analyzing complex systems using the PN formalism and its extensions. In particular, for the purposes of the framework presented in this paper, the GreatSPN GUI has been upgraded to support the Extended Stochastic Symmetric Net (ESSN), a high-level Petri Net formalism, which enables users to define a system in a compact and parametric manner and to specify in a natural manner the rate functions which may be associated with the model reactions

The other two modules, consisting of an R library and a set of docker images, implement all the framework functionalities needed for the model analysis. Docker containerization, a lightweight Operation System (OS)-level virtualization, is exploited to simplify the distribution, utilization and maintenance of the analysis tools; the R library provides an easier user interface for which no knowledge of the docker commands is needed.

The tool was successfully exploited in different contexts ranging from COVID-19 [21] to Multiple Sclerosis [22, 23, 24] and Pertussis [25].
Nevertheless, in these studies, the limitations of the current data-sensitivity analysis, optimization and data fitting tools integrated into the GreatMOD clearly came to light.
According to these, we believe that the improvement tools could be a case study for the FL5 working group.

Sensitivity Analysis. It allows us to identify among the input parameters which are the sensitive ones (i.e., those that have a great effect on the model behaviour). This may simplify the calibration step by reducing (1) the number of variables to be estimated and (2) the search space associated with each estimated parameter. In GreatMOD the R function sensitivity_analysis() implements the sensitivity analysis starting from a model exploiting the Partial Rank Correlation Coefficient (PRCC) analysis [26].

Model Calibration. The aim of this phase is to adjust the model input parameters to have the best fit of simulated behaviours to the real data. The model calibration is performed by the R function model calibration() which exploits different solvers based on Genetic Algorithms, Generalized Simulated Annealing, and Differential Evolution for solving non-linear optimization programs in which constraints are potentially non-linear.

UNITOV is working on Computational Methods in Molecular Dynamics simulations of biomolecules. Molecular Dynamics simulations based on classical force fields are a widespread technique in many chemical, physical, and biological research areas. The classical force fields are preferred each time the complexity of the conformational space that needs to be sampled is prohibitive for the quantum mechanical approach. This is the case, for example, in many problems of biological interest, for which the dynamics of proteins play a pivotal role. This kind of simulations is based on the ergodic hypothesis, which assumes that the average of a process parameter over time and the average over the statistical ensemble are the same. This is only true if the simulation time is long “enough”. For this reason, over the years, many efforts have been made to optimize the algorithms, mainly making large use of parallel computing, including GPU-based architectures. Nowadays, software packages are available, sometimes developed under the open-source approach, with performances challenging to approach by using homemade codes. Among others, Gromacs (www.gromacs.org/), Namd (http://www.ks.uiuc.edu/Research/namd/), Amber (https://ambermd.org/), and Charmm (https://www.charmm.org/) represent a standard for the community, thanks to their performances and their flexibility. We will use the Gromacs software package [27, 28] to perform our benchmarks. The parallelization philosophy of GROMACS is based on a domain decomposition approach, where the simulation system is divided into multiple subdomains that can be independently simulated. Each subdomain is assigned to a different processing unit allowing the simulation to be parallelized across multiple processors. GROMACS supports GPU acceleration. This is achieved using the CUDA language developed by NVIDIA. The peculiarity of GROMACS is that it is based on a Hybrid MPI/OMP thread parallelization for calculating the potential function (bonded and non-bonded parameters), which allows for retaining good performance in diverse architectures. Different parameters can be tuned to optimize the parallelization process [29]. Tools have been developed to help the optimization process in different architectures, such as MDBenchmark [30] or the more general Scalasca [31], and their usage will also be evaluated.

Section 1.4: solution of large-scale sparse linear systems in a parallel distributed and hybrid environment (MPI, OpenMP functionality, and use of GPU accelerators) (UNIPI, UNITOV, UNICT)

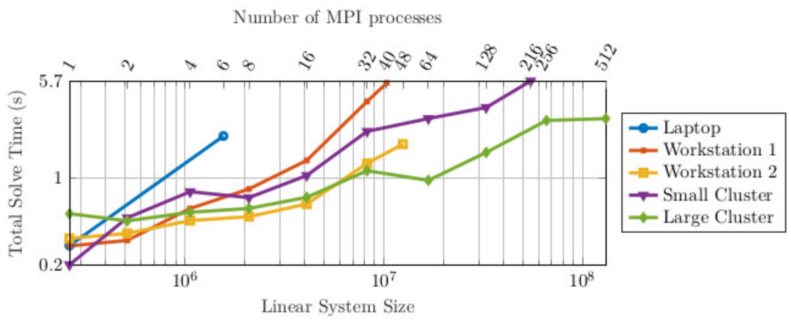
The solution of linear algebraic systems lies at the core of many scientific and engineering simulations, from the approximation of the solution and optimal controls of partial differential equations coming from engineering and physical simulations to the analysis of complex systems such as networks, queues, and other phenomena with complex interactions via, e.g., the computation of matrix functions with rational Krylov methods.

The state-of-the-art concerns the solution of linear systems with tens of billions of unknowns fully using the current pre-exascale machines. The currently existing libraries that come closest to the scale needed are Hypre, developed by Lawrence Livermore National Lab, Trilinos, developed by Sandia National Labs, they are Hypre, developed by Lawrence Livermore National Lab, Trilinos, developed by Sandia National Labs, and the AGMG Library. More recently, the development of the PSCToolkit library. More recently, the development of the PSCToolkit [33] (https://psctoolkit.github.io/) has proven to be able to tackle problems on the same size scale and with comparable performance. At the heart of all these different approaches are algorithms of the Algebraic Multigrid type (AMG). It is a class of algorithms that do not exploit the information concerning the source of the linear system to be solved, i.e. algorithms that use only the information contained in the coefficients of the system matrix. The choice made to aim at the construction of solvers that are as close as possible to being general purpose and that can be inserted into other application codes, for example, libraries for solving partial differential equations using finite elements, without requiring the modifying the code related to the other phases of the problem. The line of development in this direction is to extend the code already available in the MPI and CUDA environment to shared memory functions and, therefore, to the OpenMP framework. The goal is to have an implementation that is also optimized at the node level. We consider here an example of the tested algorithms from [34], this is the solution of a standard finite difference discretization of a Poisson problem on a 3D grid; this is a classical benchmark for AMG algorithms. To test the applicability and transversality of the PSCToolkit library, UNIPI tested the mini-app on different machines and development environments

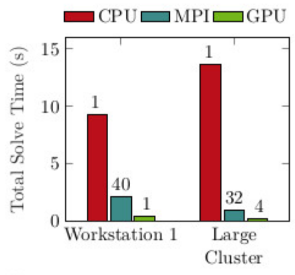
Machine	Hardware	Environment

Laptop	CPU: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz; Memory: 16Gb.	Compiler: GNU Suite 11.3.0; MPI: OpenMPI 4.1.2; BLAS: OpenBLAS 0.3.20
Workstation 1	CPU: Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz; Memory: 64Gb; GPU: NVIDIA Quadro RTX 5000.	Compiler: GNU Suite 11.3.0; MPI: mpich 3.4.3; CUDA: Cuda compilation tools, release 11.3, V11.3.109; BLAS: ATLAS 3.10.
Workstation 2	CPU: Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz; Memory: 1.48T.	Compiler: GNU Suite 6.1.0; MPI: OpenMPI 1.10.7; BLAS: OpenBLAS 0.3.3.
Small cluster (Toeplitz, Math Department UNIPI)	CPU: 1 Node (cl1) with Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz, 4 nodes (cl2) with Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz; Memory: (cl1) 126Gb, (cl2) 252Gb; Network: Intel 10-Gigabit X540-AT2.	Compiler: GNU Suite 12.2.0; MPI: OpenMPI 4.1.4; BLAS: OpenBLAS 0.3.20.
Large cluster (Marconi-100, Cineca)	CPU 980 nodes with 2×16 cores IBM POWER9 AC922 at 2.6(3.1) GHz; Memory: 256 GB/node; Network: Mellanox IB EDR DragonFly++ 100Gb/s; GPU: 4 × NVIDIA Volta V100 GPUs/node, Nvlink 2.0, 16GB.	Compiler: GNU Suite 11.2.0; MPI: OpenMPI 4.1.2; BLAS: BLAS 3.10.0; CUDA: Cuda compilation tools, release 11.6, V11.6.124.

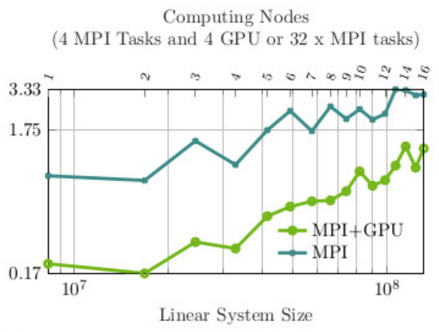
For which the scalability and performance results are given in the following figure,



(a) Weak scaling. The size per computing element is set to $n = 256000$ unknowns, and solve time is measured in seconds.



(b) Exploiting a full node, the number on top of the bars is the number of computing units.



(c) Weak scaling. The size per computing element is set to $n = 256000$ unknowns, and solve time is measured in seconds. We use from 1 to 16 nodes either with flat MPI (32 tasks per node) or using 4 MPI tasks per node with 4 GPUs.

We observe behavior in line with the expected one given the performances on a single node and the performances using
pure MPI. Small fluctuations are observed in the solution times which seem mostly attributable to the machine; in fact, the number of iterations for the use case with pure MPI is contained between 8 and 15, while in the hybrid case between 8 and 12. For results in extreme scalability (27000 MPI tasks, and 2048 GPUs) and comparison tests between the various preconditioners and the state-of-art, we refer to [33].

UNICT implemented a multigrid approach to solve the Poisson equation on MPI architectures using the PETSc library suite.
Poisson equation is central to countless applications, such as fluid dynamics, gravitational problems, electromagnetism, fluid-structure interactions.

Linear Algebra is and will continue to be at the heart of HPC applications for the foreseeable future. An immense amount of research has been devoted to the efficient implementation of linear algebra; among these efforts we can identify some that are more concerned with the computational models needed in approaching the use of linear algebra inside applications.

Among these trends UNITOV identified the emergence of the so-called task-based runtime environment [35, 36]; these systems provide a novel way to encode complex algorithms by specifying a set of dependencies among various building blocks. The programmer builds a DAG (directed acyclic graph) specifying for each node a kernel to be executed on a certain portion of the data, and connecting the nodes with directed arcs to specify an input-output relationship among the various computations. The end result is the increase in programmability of various kinds of complex linear algebra algorithms.

As we mentioned previously, linear algebra has elicited an immense amount of research work, due to its importance in providing application building blocks; and yet, there is a certain amount of disconnection between the users and programmers of applications, and the developers of libraries. The libraries encompass a body of knowledge on what constitutes efficient implementations, but the users tend to rely ever more on environments that may or may not provide an optimal mapping from problem to function calls. As noted in the survey [41], the mapping problem itself is NP-complete, hence there is a need for further activity in this field to help users identify the best possible ways to frame the applications in ways that are conducive to exploitation of exascale resources.

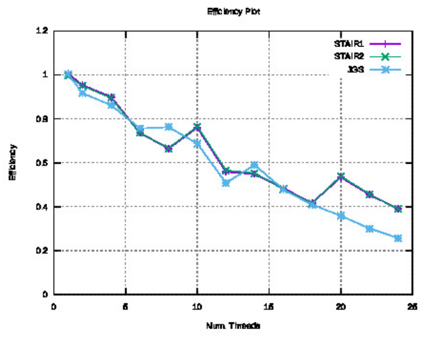
One of the essential ingredients of modern HPC architectures is their heterogeneity; handling heterogeneity in the applications has been addressed e.g. by using the techniques in [39, 40], but a lot more work is needed in training end-users on how to enable heterogeneity.

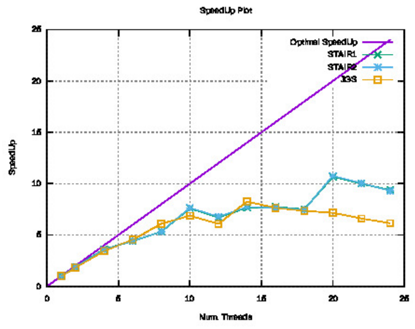
In the recent past UNITOV has introduced new versions of their library software for sparse linear algebra on high performance computers [37, 38], where they implemented some among the most effective solver techniques available, i.e. algebraic multigrid preconditioners coupled with Krylov subspace solvers.
UNITOV recent research has been focused on the implementation of more effective strategies for building the multigrid hierarchy our research program has been accepted for early access to the new Leonardo computational facility at CINECA, where UNITOV plans to explore extreme scalability of multigrid construction based on graph matching [37, 38].

Section 1.5: Solution of structured linear systems coming from the study of Markov chains (UNIPI)

Commentato [AO2]: Dovrebbe essere UNIPI

The focus is on the computation of the stationary vector for a continuous-time Markov chain. Such discrete-state models are widely employed for modelling and analysis of large networks and systems such as communication networks, allocation schemes, computer systems, and population processes. To compute the quantity of interest one has to solve a homogeneous linear system. Again, due to the scale of the problem, the only possible approach is to use iterative methods that can be implemented in an HPC environment. The state of the art in this direction concerns the use of simple fixed point methods based on the splitting of the system matrix. The promising direction in which to develop methods and algorithms is that of being able to exploit the structure of the matrix in order to obtain scalability in a parallel environment [32]. The current approach allows solving problems of this form in a shared memory environment (OpenMP), with the prospect of extending the approach from a node-level optimization to one that can actually be used in a distributed environment, possibly in a hybrid way. A strong scaling example of the tested algorithms from [32] run on a single with two Intel Xeon E5-2650v4 CPUs with 12 cores and 24 threads each, running at 2.20GHz (without Hyper-threading to have a number of logical processors equal to physical processors) of the three versions of the algorithm exploiting the new structured approach on a standard benchmark is given in the following figure.





We observe that there is still room for improvement with respect to the optimal achievable efficiency (consider the violet line).

Section 1.6: benchmarking and performance modelling of accelerators; algorithms for tensor cores and processing-in-memory; memory management in multi-processor architectures (UNIPD)

The widespread interest in machine learning applications has boosted the introduction of hardware accelerators. Since the boost is market-driven, each silicon producer touts exceptional speed or power efficiency, depending on the application, but, to the best of our knowledge, there is little scientific work comparing solutions from different vendors systematically and quantitatively, regardless of the metric. Performance figures are reported, but they are customarily specific for a single accelerator and software stack. Reports on factors influencing performance tend to be based on bottleneck analysis and can not be used for predictions. On the other end of the spectrum, predictive register-level models (e.g., https://doi.org/10.1109/MM.2020.2985963, https://doi.org/10.1145/3466752.3480063) have been proposed to explore the hardware design space. Still, they are not useful for high-level decisions on the structure of the computation, algorithms, or the best accelerator for a given performance metric.

Recent works have addressed how to include within the algorithm design process the features of modern hardware accelerators like tensor cores and processing-in-memory. For instance, tensor cores have been used for accelerating important primitives as scan operations, linear algebra, similarity search (e.g. [46, 45, 44]). Processing-in-memory architectures have been used for accelerating data structures as skip lists or irregular computations line skyline computations (e.g., [47, 50]).

Memory management in the presence of multiple instruction streams, each due to one processor in a multi-processor architecture, has been extensively studied, particularly after multi-core processors became mainstream. In this scenario, the memory is shared among p possibly heterogeneous processors, each running, independently and concurrently, its own program. The algorithmic decision faced by the memory-management policy is the same as in the sequential case, that is, deciding which page to evict when a new page has to be brought into fast memory; note that these decisions effectively determine a
partition of the memory among processors that changes over time. The objective is to design a replacement policy that minimizes some objective function of processors' completion times. This problem has also received considerable attention, starting from some pioneering work, such as [49], on (offline and online) heuristics that dynamically adjust the sizes of the memory partitions dedicated to each processor; it resisted a rigorous theoretical characterization until recently when the first algorithmic solutions, as well as fundamental limits, have been shown under standard worst-case analysis [48, 42, 43]. Future directions of research include: randomized parallel paging policies with better guarantees than those achievable by deterministic policies [42, 43], or a proof that this is not possible (i.e., that randomization does not help in parallel paging); parallel paging policies that use (machine-learning) predictions on future page requests: we aim for policies that have near-optimal performance when these predictions are accurate, but recover the prediction-less worst-case behavior when the predictions have large errors.

Section 1.7: Cellular Automata (CA) models (UNICAL)
The main activities of UNICAL are focused on OpenCAL++, a platform for transparent and efficient parallel execution of Cellular Automata (CA) models [4], and on the development of Load Balancing techniques for CA execution by devising simple closed-form expressions that allow to compute the optimal workload assignment in a dynamic fashion, with the goal of guaranteeing a fully balanced workload distribution during the parallel execution. The Cellular Automata (CA) computational paradigm can be easily adopted to model and simulate complex systems characterized by a high number of interacting elementary components. Due to their implicit parallel nature, CAs can be productively parallelized across multiple parallel machines to scale and speed up their execution. Execution of CA on both sequential and parallel computers consists in a step-by-step evaluation of the transition function for each cell of the cellular space.
Even though Parallel Computing has undoubtedly in general proved its effectiveness in many CA application scenarios, overheads can arise due to the parallelization process itself, that can reduce the obtainable benefits [51-53]. To reduce this overhead, different strategies have been envisioned [54, 55]. Moreover, the low-level implementations of CA execution must be devised for each parallel execution context, such as shared memory (e.g., OpenMP) and distributed memory (e.g., MPI) architectures and modern GPGPU (e.g., CUDA or OpenCL). The choice of the suitable execution context based on hardware availability is a key factor for providing dramatic computational improvements in computational results. On the other side, parallel programming requires strong technical expertise, let alone considering the different parallel execution contexts and adopted optimization strategies [54]. Indeed, different high level CA modelling APIs were proposed in the literature in attempting to mitigate these issues. However, in general, these solutions

lack a full portability across different execution contexts and parallelization strategies requiring the model program to be adapted from case to case (e.g., [56]).
For this aim, the OpenCAL++ makes parallelism transparent to the modeler and addresses many aspects of the underlying formal computational paradigms and optimization strategies. Moreover, it implements a set of load balancing strategies to accelerate the computation. The adoption of a fully object-oriented approach enables the full transparency of the code and a plug-and-play feature that allows to easily insert new parallel optimization strategies and new parallel execution contexts.

A second activity will regard Load Balancing (LB) techniques, which are widely adopted in many scientific contexts with the aim of improving the computational performances of CA models when executed on a parallel system. LB techniques fall into two main classes, namely static and dynamic load balancing (cf. [58-62]). Static load balancing strategies can be further classified based on how the domain is partitioned: Boxwise, in the case of row-wise decomposition, Stripwise, in the case of column-wise decomposition and Scattered where chessboard partitioning is adopted. As an example of static LB, in the partitioning of a large scale urban simulation is initially determined before execution with the aim of avoiding that large non-urbanizable areas could be allocated in the same partitions thus resulting in a workload unbalance. In principle, dynamic load balancing can potentially outperform static ones as it aims to balance the workload among processors during the execution, with the consequence of better adapting to the unpredictable changes of the workload distribution. Dynamic load balancing strategies can be classified in Diffusion, Dimension Exchange and Gradient approaches [63, 64]. Diffusion is a highly distributed local approach that uses near-neighbor load information to distribute excess load from the more loaded processors to the nearby under loaded ones; in Dimension Exchange, LB is performed iteratively by reducing an N processor system to a log N dimensions one for the purpose of balancing one dimension at a time; in the Gradient model, a gradient map of underloaded processors in the system is used to migrate tasks between overloaded and underloaded processors.

Section 1.8: Locality Exploitation in High Performance Speculative Simulation on Shared-Memory Machines (UNITOV)

Shared-memory multi-processor/multi-core machines have become extremely attractive for innovative Parallel Discrete Event Simulation (PDES) techniques where the workload of the events to be processed is fully shared among all the CPU units. In this scenario, aspects like locality and its impact on the effectiveness of the hardware level configuration (cache hierarchy and NUMA nodes) is becoming a critical factor for scaled up performance along the path to exascale computing [64].

In this area, a few works investigated the management of buffers [65] or the reduction of the number of message copies to exchange data [66]. The work in [67] discusses how the setup of intra-node facilities---based on shared-memory---can allow multiple MPI ranks running on a same machine to reduce their communication overhead. The proposal in [68] introduces an architecture for speculative PDES where multiple threads running on a shared-memory machine support communication based on top/bottom half primitives with reduced intrusiveness. Additional studies have been focused on the analysis of general architectural redesigns when moving (speculative) PDES to shared-memory/multi-core machines (see [69, 70]).

As for solutions oriented to load sharing in multi-core machines, in combination with long-term binding of simulation objects to worker threads, we can find model-based approaches. They allow the calculation of an effective distribution of simulation objects among threads under the hypothesis that the future of the simulation run will have similarities with respect to the last observed execution phase [71] For this scenario, the literature also offers approaches where the state of the simulation object is dynamically migrated across the different NUMA nodes in the shared-memory machine [72] to make the in charge worker thread (after the rebalance) more effective in the access to state information.

In [73] the authors exploit a fully shared event pool in order to enable any worker thread to CPU-dispatch any simulation object at any time along the simulation execution. Hence, a simulation object is kept locked by a specific worker thread only for the time interval related to the processing of an individual event. In this work, the accesses to the shared event pool have been based on a non-blocking algorithm, which favors scalability. The major limits of this work are related to the fact that spatial locality is not taken into account. Hence, a worker thread can continuously switch across different simulation objects, with no attempt to reuse the same memory areas (keeping the state of specific simulation objects) which have been accessed recently. Also, the accesses are NUMA unaware, hence they can generate both delay---because of the latency for accessing far NUMA nodes---and excessive pressure on the NUMA interconnection, limiting performance and scalability. As we pointed out, these are the baseline problems we tackle in this article, in combination with the reduction of the amount of memory locations accessed by a worker thread when managing the shared event pool.

The work in [74] provides an improvement for load-balancing in PDES on top of shared-memory multi-processor/multi-core machines. It combines a classical medium/long-term binding scheme based on persistence---namely, past data related to the workload---and work-stealing. The stealing operation is put in place if the last re-balance has led to imperfect partitions---this may occur because of errors in the prediction of the future workload that will be actually generated by the simulation objects, when the re-balance occurs. In this solution the simulation objects are still grouped and remain bound to a specific worker thread, up to some steal operation or a periodic re-balance. Hence the core ideas in this proposal are still not suited for locality improvement with fine-grain sharing of the workload among worker threads. Additionally, in this proposal the authors do not tackle spatial locality, hence they do not explicitly address the improvement of cache and NUMA usage.

The work in [75] provides a solution for improving the efficiency of cache usage in speculative PDES systems. This solution is based on redirecting cache-adverse operations---like checkpointing, which leads to the invalidation of other information kept into the cache---to a specific cache partition. In practice, it offers the advantage of keeping some zone of

the cache less affected by cache replacement caused by write intensive operations related to data that are likely not re-accessed for a while (like it occurs for a checkpoint).
The work in [76] provides an analysis showing how checkpointing and reverse computing have different impacts on speculative PDES performance on a shared-memory machine because of their different effects on the TLB architecture.
These proposals are however not tailored for the optimization of cache management in the context of fine-grain sharing of simulation objects among worker threads, and are also not tailored to the improvement of the effectiveness of memory accesses in NUMA architectures when cache misses occur. These aspects are instead central for our proposal.

Our research plan differs from these studies since none of them is oriented to the combination of 1) cache-aware association of simulation objects to threads, 2) NUMA-aware placement/access of/to simulation objects' states and 3) batch-processing of the events of specific simulation objects along any thread. As a matter of fact, our research plan has relations with classical mechanisms that have been used in operating systems in order to CPU-schedule the different threads. In more detail, the perfect load sharing approach, which has been used in Linux (see https://mirrors.edge.kernel.org/pub/linux/kernel/v2.4/), enables a same thread to consume its residual ticks according to a batching scheme. This enables the thread to exploit the caching system in a more effective manner---compared to the scenario where multiple threads still having ticks to spend on the CPU are dispatched in an alternate manner. We plan to exploit a kind of batch-processing for enabling a worker thread to process events of the simulation objects reducing the alternance. However, our solution also needs to take into account aspects that are not considered at the level of the operating system technology, like the need for avoiding timestamp order violations as much as possible, in order to reduce the incidence of wasted computation and rollbacks in the parallel run.

Section 2: Cooperations between groups

In this section we summarize some of the connections that have been created or strengthened within the project. Most connections have already been detailed in Section 1, therefore we provide a brief sketch here. We refer to the connections between units of the project, as well as with external research groups.

INAF collaborates with UNICT, UNITO, CINECA.

UNIPD collaborates with:
ETH
IT Univ. of Copenhagen
with Massimo Bernaschi, CNR, on the development of a GPU-accelerated sparse matrix times sparse matrix kernel for distributed memory systems able to scale to hundreds of compute nodes by overlapping computation and communication

Regarding UNIPI, part of the survey work on the algorithms for the solution of sparse linear systems and on the implementation of the new functions of the PSCToolkit library was carried out in collaboration with the research unit at the University of Rome "Tor Vergata".

IIT cooperates with other groups within the scope of Spoke 1:
MOX POLIMI (Prof. De Falco)
UNITO (Prof. Aldinucci)
UNIPI (Prof. Ferragina)

UNIFE cooperates with:
RWTH Aachen University (Prof. Michael Herty, collaboration on optimization method and applications to traffic flows)
TU Munich (Prof. Massimo Fornasier, collaboration on optimization methods and machine learning)
U. Lille (Prof. Thomas Rey, collaboration on numerical simulations of the Boltzmann equation)
U. Toulouse (Prof. F. Filbet, collaboration on numerical methods in plasma physics)
GSSI L'Aquila (Prof. N. Guglielmi, collaboration on numerical methods in biomedical and epidemiological applications)
U. Catania (Prof. G. Russo, collaboration on numerical methods for kinetic equations with multiple scales)

UNICAL cooperates with:
UPC Barcelona TECH – Spain
BSC Supercomputing Center – Barcelona - Spain
ICAR – CNR - Italy

Section 3: Reference list

Reference list

1. Sciacca, Eva, et al. "An integrated visualization environment for the virtual observatory: Current status and future directions." Astronomy and Computing 11 (2015): 146-154.

2. Vitello, F., et al. "Vialactea visual analytics tool for star formation studies of the galactic plane." Publications of the Astronomical Society of the Pacific 130.990 (2018): 084503.

3. Vecchiato, Alberto, et al. "The global sphere reconstruction for the gaia mission in the astrometric verification unit." Software and Cyberinfrastructure for Astronomy II. Vol. 8451. SPIE, 2012.

4. Cesare, Valentina, et al. "The Gaia AVU–GSR parallel solver: Preliminary studies of a LSQR-based application in perspective of exascale systems." Astronomy and Computing 41 (2022): 100660.

5. Springel, Volker, Naoki Yoshida, and Simon DM White. "GADGET: a code for collisionless and gasdynamical cosmological simulations." New Astronomy 6.2 (2001): 79-117.

6. Mignone, Andrea, et al. "PLUTO: a numerical code for computational astrophysics." The Astrophysical Journal Supplement Series 170.1 (2007): 228.

7. Mignone, Andrea, et al. "The PLUTO code for adaptive mesh computations in astrophysical fluid dynamics." The Astrophysical Journal Supplement Series 198.1 (2011): 7.

8. M. Volpi, U. Locatelli, M. Sansottera.: A reverse KAM method to estimate unknown mutual inclinations in exoplanetary systems. Cel. Mech. \& Dyn. Astr., 130:36 (2018).

9. C. Caracciolo, U.Locatelli, M.~Sansottera, M.~Volpi.: Librational KAM tori in the secular dynamics of the upsilon-Andromedae planetary system. MNRAS, 510, 2147--2166 (2022).

10. A. Giorgilli, M. Sansottera.: Methods of algebraic manipulation in perturbation theory. In P.M. Cincotta, C.M. Giordano, C. Efthymiopoulos (eds.): “Chaos, Diffusion and Non-integrability in Hamiltonian Systems --Applications to Astronomy”, Proceedings of the Third La Plata International School on Astronomy and Geophysics, Universidad Nacional de La Plata and Asociacion Argentina de Astronomia Publishers, La Plata (2012).

11. U. Locatelli, C.Caracciolo, M. Sansottera, M.Volpi.: A numerical criterion evaluating the robustness of planetary architectures; applications to the upsilon-Andromedae system, In A. Celletti, C. Galeş, C. Beaugé, A. Lemaitre, eds., Multi-scale (time and mass) dynamics of space objects, Proceedings of the International Astronomical Union Symposium No. 364, Book Series, Volume 15, 65--84 (2021).

12. Amestoy P.R., Duff I.S., L'Excellent J.-Y., Koster J., MUMPS: a general purpose distributed memory sparse solver International Workshop on Applied Parallel Computing, Springer (2000), pp. 121-130
13. Ghysels P., Synk R., High performance sparse multifrontal solvers on modern GPUs, Parallel Computing 110 (2022) pp. 102897.
14. Sumikoshi K. et al., "A Fast Protein-Protein Docking Algorithm Using Series Expansion in Terms of Spherical Basis Functions", Genome Informatics,16(2): 161–173 (2005).
15. S.-Y. Huang, "Search strategies and evaluation in protein–protein docking: principles, advances and challenges", Drug Discovery Today, 19(8), 1081-1096, (2014).
16. Friedrich Förster, "Subtomogram analysis: The sum of a tomogram's particles reveals molecular structure in situ", J. Structural Biology, X, 6, 100063, 2022.
17. A.S. Frangakis et al., "Identification of macromolecular complexes in cryoelectron tomograms of phantom cells", PNAS, 99 (22) 14153-14158
18. S. Decherchi et al., "Between algorithm and model: different Molecular Surface definitions for the Poisson-Boltzmann based electrostatic characterization of biomolecules in solution", Comm. Comp. Phys., 13(1), 61-89, 2013
19. L. Gagliardi and W. Rocchia, "SiteFerret: beyond simple pocket identification in proteins", arXiv preprint arXiv:2212.11888, 2022

20. GreatSPN enhanced with decision diagram data structures. Babar, J., Beccuti, M., Donatelli, S., Miner, A.S. In: Application and Theory of Petri Nets.PETRI NETS 2010. LNCS, vol. 6128, pp. 308–317 (2010)

21. Impacts of reopening strategies for COVID-19 epidemic: a modeling study in Piedmont region. S. Pernice, P. Castagno, L. Marcotulli, M. M. Maule, L. Richiardi, G. Moirano, M. Sereno, F. Cordero and M. Beccuti. BMC Infectious Diseases, Volume 20, Article number: 798 (2020).

22. Multiple Sclerosis disease: a computational approach for investigating its drug interactions. S. Pernice, M. Beccuti, G. Romano, M. Pennisi, A. Maglione, S. Cutrupi, F. Pappalardo, L. Capra, G. Franceschinis, M. De Pierro, G. Balbo, F.Cordero and R. Calogero. Proceeding of 16th Int. Conference Computational Intelligence methods for Bioinformatics and Biostatistics (CIBB 2019), volume 12313 LNBI, pp. 299-308, Bergamo, Italy, September 4-6, 2019.

23. Estimating Daclizumab effects in Multiple Sclerosis using Stochastic Symmetric Nets. S. Pernice, M. Beccuti, P. Do', M. Pennisi, and F. Pappalardo. In 2nd Int. Workshop on Computational Methods for the Immune System Function (CMISF 2018), December 3 - 6, 2018, Madrid, Spain, Article number 8621259, Pages 1393-1400.

24. Exploiting Stochastic Petri Net formalism to capture the Relapsing Remitting Multiple Sclerosis variability under Daclizumab administration. S. Pernice, G. Romano, G. Russo, M. Beccuti, M. Pennisi, and F. Pappalardo. In 3rd Int. Workshop on Computational Methods for the Immune System Function (CMISF 2019), San Diego, USA, November 2019.

25. A computational framework for modeling and studying pertussis epidemiology and vaccination. P. Castagno, S. Pernice, G. Ghetti, M. Povero, L. Pradelli, D. Paolotti, G. Balbo, M. Sereno and M. Beccuti. BMC Bioinformatics, Volume 21, 16 September 2020, Page 344.

26. A methodology for performing global uncertainty and sensitivity analysis in systems biology. Marino, S., Hogue, I.B., Ray, C.J., Kirschner, D.E. Journal of Theoretical Biology 254(1), 178–196 (2008)

27. S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, E. Lindhal (2020). Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. The Journal Of Chemical Physics, 153(13), 134110. doi: 10.1063/5.0018516
28. S. Pronk, S. Páll, R. Schulz, P. Larsson, P., Bjelkmar, R. Apostolov, et al. (2013). GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. Bioinformatics, 29(7), 845-854. doi: 10.1093/bioinformatics/btt055
29. C. Li, W. Chen, Y, Zhang, Q. Bai . Analyses on Performance of Gromacs in Hybrid MPI+OpenMP+CUDA Cluster. (2014)., 2014 IEEE International Conference on High Performance Computing and Communications (HPCC), 2014 IEEE 6th International Symposium on Cyberspace Safety and Security (CSS) and 2014 IEEE 11th International Conference on Embedded Software and Systems (ICESS). doi 10.1109/HPCC.2014.157
30. M. Gecht, M. Siggel, M. Linke, G. Hummer, J. Köfinger MDBenchmark: A toolkit to optimize the performance of molecular dynamics simulations. J. Chem. Phys. 153, 144105 (2020); doi 10.1063/5.0019045
31. M. Geimer, F. Wolf, B. J.N. Wylie , E. Abraham , D. Becker , B. Mohr. The Scalasca performance toolset architecture. (2010). Concurrency Computat.: Pract. Exper. 22, 702–719; doi 10.1002/cpe.1556.
32. V. Besozzi; M. Della Bartola; L. Gemignan; Experimental Study of a Parallel Iterative Solver for Markov Chain Modeling. International Conference on Computational Science, ICCS-2023. Springer (2023). In press
33. D'Ambra, P.; Durastante, F.; Filippone, S.. AMG Preconditioners for Linear Solvers towards Extreme Scale. SIAM J. Sci. Comput. 43 (2021), no. 5, S679-S703. MR4331965
34. D’Ambra, P.,; Durastante F.; Filippone, S.. Parallel Sparse Computation Toolkit" Software Impacts (2023): 100463.

35. Emmanuel Agullo, Olivier Aumage, Mathieu Faverge, Nathalie Furmento, Florent Pruvost, Marc Sergent, and Samuel Paul Thibault. 2017. Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model. IEEE

Transactions on Parallel and Distributed Systems (2017), 1–1. https://doi.org/10.1109/TPDS.2017.2766064

36. Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche, Julien Herrmann, and Antoine Jego. 2023. Task-Based Parallel Programming for Scalable Matrix Product Algorithms. ACM Trans. Math. Softw. (feb 2023). https://doi.org/10.1145/3583560

37. P. D'Ambra, F. Durastante, S. Filippone AMG Preconditioners for Linear Solvers towards Extreme Scale SIAM J. Sci. Comput., SPECIAL SECTION Copper Mountain 2020, Vol. 43, No. 5, 2021, pp. S673-S703

38. P. D'Ambra, F. Durastante, and S. Filippone: Parallel Sparse Computation Toolkit Software Impacts 2023,

39. Salvatore Filippone, Valeria Cardellini, Davide Barbieri and Alessandro Fanfarillo: Sparse Matrix-Vector Multiplication on GPGPUs ACM Transactions on Mathematical Software (TOMS), Volume 43 Issue 4, December 2016, Article No. 30,

40. V. Cardellini, S. Filippone, D. Rouson: Design Patterns for sparse-matrix computations on hybrid CPU/GPU platforms. Scientific Programming, 22 (2014), pp. 1-19.

41. Christos Psarras, Henrik Barthels, and Paolo Bientinesi. 2022. The Linear Algebra Mapping Problem. Current State of Linear Algebra Languages and Libraries. ACM Trans. Math. Softw. 48, 3, Article 26 (sep 2022), 30 pages. https://doi.org/10.1145/3549935

42. Kunal Agrawal, Michael A. Bender, Rathish Das, William Kuszmaul, Enoch Peserico and Michele Scquizzato. Tight bounds for parallel paging and green paging. Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), 2021.
43. Kunal Agrawal, Michael A. Bender, Rathish Das, William Kuszmaul, Enoch Peserico and Michele Scquizzato. Online parallel paging with optimal makespan. Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2022.
44. Thomas D. Ahle and Francesco Silvestri. Similarity Search with Tensor Core Units. In Proc. 13th International Conference on Similarity Search and Applications (SISAP), 2020.
45. Rezaul Chowdhury, Francesco Silvestri and Flavio Vella. Algorithm design for Tensor Units. In Proc. 27th International European Conference on Parallel and Distributed Computing (EURO-PAR), 2021. Extended version of SPAA 2020 Brief Announcement
46. Abdul Dakkak, Cheng Li, Jinjun Xiong, Isaac Gelado, and Wen-mei Hwu. Accelerating reduction and scan using tensor core units. In Proceedings of the ACM International Conference on Supercomputing (ICS '19). 2019
47. Hongbo Kang, Phillip B. Gibbons, Guy E. Blelloch, Laxman Dhulipala, Yan Gu, and Charles McGuffey. 2021. The Processing-in-Memory Model. In Proceedings of the

33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '21). Association for Computing Machinery, New York, NY, USA, 295–306, 2021.

48. Michele Scquizzato. Paging on Complex Architectures. PhD Thesis, University of Padova, 2013.
49. Harold S. Stone and John Turek and Joel L. Wolf. Optimal partitioning of cache memory. IEEE Transactions on Computers, 1992.
50. Vasileios Zois, Divya Gupta, Vassilis J. Tsotras, Walid A. Najjar, and Jean-Francois Roy. Massively parallel skyline computation for processing-in-memory architectures. In Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques (PACT '18), 2018.
51. F. Cicirelli, A. Forestiero, A. Giordano, and C. Mastroianni, “Parallelization of space-aware applications: Modeling and performance analysis,” Journal of Network and Computer Applications, vol. 122, pp. 115–127, 2018.
52. J. Was, H. Mr´oz, and P. Topa, “Gpgpu computing for microscopic simulations of crowd dynamics,” Computing and Informatics, vol. 34, no. 6, pp. 1418–1434, 2016.
53. Gerakakis, P. Gavriilidis, N. I. Dourvas, I. G. Georgoudas, G. A. Trunfio, and G. C. Sirakoulis, “Accelerating fuzzy cellular automata for modeling crowd dynamics,” Journal of Computational Science, vol. 32, pp. 125–140, 2019.
54. Giordano, A. De Rango, D. D’Ambrosio, R. Rongo, and W. Spataro, “Strategies for parallel execution of cellular automata in distributed memory architectures,” in 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE, 2019, pp. 406–413.
55. Giordano, D. D’Ambrosio, A. De Rango, A. Portaro, W. Spataro, and R. Rongo, “Exploiting distributed discrete event simulation techniques for parallel execution of cellular automata,” in Artificial Life and Evolutionary Computation, F. Cicirelli, A. Guerrieri, C. Pizzuti, A. Socievole, G. Spezzano, and A. Vinci, Eds. Cham: Springer International Publishing, 2020, pp. 66–77.
56. M. Cannataro, S. Di Gregorio, R. Rongo, W. Spataro, G. Spezzano, and D. Talia, “A parallel cellular automata environment on multicomputers for computational science,” Parallel Computing, vol. 21, no. 5, pp. 803–823, 1995.
57. Giordano, A. De Rango, D. Spataro, D. D’Ambrosio, C. Mastroianni, G. Folino, and W. Spataro, “Parallel execution of cellular automata through space partitioning: the landslide simulation sciddicas3-hex case study,” in 2017 25th Euromicro International Conference on Parallel, Distributed and Networkbased Processing (PDP). IEEE, 2017, pp. 505–510
58. R. V. Hanxleden and L. R. Scott, “Load balancing on message passing architectures,” J. Parallel Distrib. Comput., vol. 13, no. 3, pp. 312–324, 1991.
59. X. Li, X. Zhang, A. Yeh, and X. Liu, “Parallel cellular automata for large-scale urban simulation using load-balancing techniques,” Int. J. Geographical Inf. Sci., vol. 24, no. 6, pp. 803–820, 2010.
60. M. Cannataro, S. Di Gregorio, R. Rongo, W. Spataro, G. Spezzano, and D. Talia, “A parallel cellular automata environment on multi-computers for computational science,” Parallel Comput., vol. 21, no. 5, pp. 803–823, 1995.
61. D. M. Nicol and J. H. Saltz, “An analysis of scatter decom-position,” IEEE Trans. Comput., vol. 39, no. 11, pp. 1337–1345, Nov. 1990.

62. G. Cybenko, “Dynamic load balancing for distributed memory mul-tiprocessors,” J. Parallel Distrib. Comput.,vol.7,no.2,pp.279–301, 1989.
63. F. C. H. Lin and R. M. Keller, “The gradient model load balancing method,” IEEE Trans. Softw. Eng., vol. SE-13, no. 1, pp. 32–38, Jan. 1987.

64. D. A. Reed and J. J. Dongarra: “Exascale computing and big data”, Commun. ACM, Vol 58, No 7, 2015, https://doi.org/10.1145/2699414

65. R. Fujimoto and K. Panesar: “Buffer Management in Shared-Memory Time Warp Systems”, Proc. of Ninth Workshop on Parallel and Distributed Simulation, 1995, https://doi.org/10.1145/214282.214330

66. B. Swenson and G. Riley: “A New Approach to Zero-Copy Message Passing with Reversible Memory Allocation in Multi-Core Architectures”, Proc. of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, 2012, https://doi.org/10.1109/PADS.2012.3

67. C.J. Ross, C. D. Carothers, M.Mubarak, R. B. Ross, J. Kelvin Li and K. Ma: “Leveraging Shared Memory in the Ross Time Warp Simulator for Complex Network simulations”, Proc. of the 2018 Winter Simulation Conference, {WSC} 2018, https://doi.org/10.1109/WSC.2018.8632333.

68. R.Vitali, A.Pellegrini and F. Quaglia: “Towards Symmetric Multi-threaded Optimistic Simulation Kernels”, Proc. of the 26th {ACM/IEEE/SCS} Workshop on Principles of Advanced and Distributed Simulation, PADS 2012, https://doi.org/10.1109/PADS.2012.46

69. R. Fujimoto: “Time Warp on a Shared Memory Multiprocessor”, Proc.of the International Conference on Parallel Processing, IICPP '89, 1989

70. J. Wang, D. Jagtap, N. Abu-Ghazaleh and D. Ponomarev: “Parallel discrete event simulation for multi-core systems: Analysis and optimization”, IEEE Transactions on Parallel and Distributed Systems, Vol 25, no 6, pp 1574-1584, https://doi.org/10.1109/TPDS.2013.193

71. R. Vitali, A.Pellegrini and F. Quaglia: “Load sharing for optimistic parallel simulations on multi core machines”, ACM SIGMETRICS Performance Evaluation Review, Vol. 40, no. 3, 2012, https://doi.org/10.1145/2425248.2425250

72. A. Pellegrini and F. Quaglia:”NUMA Time Warp”, Proc. of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation,2015, https://doi.org/10.1145/2769458.2769479

73. M. Ianni, R. Marotta, D. Cingolani, A. Pellegrini, F. Quaglia:”The Ultimate Share-Everything PDES System”, Proc.of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, 2018, https://doi.org/10.1145/3200921.3200931

74. T. Wenjie, Y. Yiping, L. Tianlin, S. Xiao and Z. Feng:”An Adaptive Persistence and Work-stealing Combined Algorithm for Load Balancing on Parallel Discrete Event Simulation”, ACM Trans. Model. Comput. Simul., Vol. 30, No. 2, 2020, https://doi.org/10.1145/3364218

75. R. Vitali, A.Pellegrini, G.Cerasuolo: ”Cache-aware memory manager for optimistic simulations”, ICST Conference on Simulation Tools and Techniques, SIMUTOOLS '12, 2012, https://doi.org/10.4108/icst.simutools.2012.247766

76. C. D. Carothers, K.S. Perumalla and R.Fujimoto: “The effect of state-saving in optimistic simulation on a cache-coherent non-uniform memory access architecture”, Proc.of the 31st conf. on Winter simulation, 1999, Phoenix, AZ, USA, https://doi.org/10.1109/WSC.1999.816902

77. “Interacting Multiagent Systems:Kinetic equations and Monte Carlo methods”, by L. Pareschi, G. Toscani, Oxford University Press (2013)

78. “Survey of multifidelity methods in uncertainty propagation, inference, and optimization” by B. Peherstorfer, K. Willcox, M. Gunzburger, SIAM Review (2018)

79. “Uncertainty quantification for hyperbolic equations” by S. Jin, L. Pareschi (Eds), in SEMA-SIMAI Springer Series (2018)