Horizon 2020 RIA: FET-HPC
Project number 671553

ExaNeSt – European Exascale System Interconnect and Storage
http://www.exanest.eu

Deliverable **D2.1**

Census of the Applications

version 1.2 – 16 June 2016

EDITOR, CONTRIBUTORS

Partner	*Authors*
eXact lab	Lead Editor: Stefano Cozzini Other Contributors: Stefano Piani
INAF	Giuliano Taffoni, Giuseppe Murante
INFN	Elena Pastorelli, Pier Stanislao Paolucci, Andrea Biagioni, Michele Martinelli, Piero Vicini
MonetDB Solutions	Ying Zhang, Niels Nes, Panagiotis Koutsourakis, Richard Koopmanschap, Martin Kersten
Allinea	Daniel Everett
Enginsoft	Gino Perna

REVISION HISTORY

Version	*Date*	*Description*
0.1	01 Feb 2016	First skeleton of the document. INFN and MonetDB contribution taken from ExaNeSt wiki
0.2	08 Feb 2016	Added eXact-lab and INAF contribution on applications. Added a comparison session.
0.3	14 Feb 2016	Added contribution from Enginsoft
0.4	15 Feb 2016	Added contribution from INFN on LQCD application
0.5	16 Feb 2016	Further contribution from INAF (on application and first part) and a first general revision by editor
0.6	22 Feb 2016	Incorporated suggestion/comments given by internal reviewers
0.7	24 Feb 2016	Incorporated some further modifications by MonetDB and INAF
0.8	25 Feb 2016	Final check by S. Cozzini as editor
0.9	26 Feb 2016	Indicated partner’s contribution in each section as suggested by Project Manager
1.0	29 Feb 2016	Official version, delivered to the European Commission.
1.1	12 Apr 2016	Same as version 1.0, but made Publicly available.
1.2	16 Jun 2016	Added references for LQCD paragraph

TABLE OF CONTENTS

LIST OF ABREVIATIONS

HPC	High Performance Computing – section 1
MPI	Message Passing Interface – section 2.1
GCM	Global Climatic Model- section 2.2
RCM	Regional Climate Model section 2.2
QCD	Quantum Chromo Dynamics – section 2.3.1
CFD	Computational Fluid Dynamics– section 2.5.1
ACID	Atomicity, Consistency, Isolation, Durability– section 3
OLAP	OnLine Analytical Processing– section 3
GIS	Geographical Information System– section 3

SUMMARY

In this document, a survey of scientific and technical applications that could be potentially used in the ExaNeSt project are presented and discussed. The applications are playing a central role in ExaNeSt: they are used to identify a set of initial requirements for the design of the platform, and then they are also used to test the infrastructure during all the implementation phases.

The selected applications represent the state of the art of HPC computing in different disciplines: Astrophysics, Material Science, Climatology, Neuroscience, etc.

The applications described in this document are a preliminary list of software and tools that will be potentially ported and re-engineered during the subsequent phases of the projects. A subset of them will be then identified.

A brief introduction discusses the guidelines used to select the applications. Each application is then presented in detail and all the technical aspects are also outlined. Finally a comparison among them is conducted.

1. Introduction

This deliverable contains a census of the scientific and technical applications that will be potentially used in the ExaNeSt project.

1.1 Motivations

The platform developed within this project, as well as the prototype hardware that we will build, will be tested with real scientific applications, that will play two roles: *(i)* identify a set of initial requirements to drive the initial development of the infrastructure, and *(ii)* test it during the implementation. In this way our platform will have the ability to run a wide array of applications, extending HPC's reach from its roots in modeling and simulation of complex physical systems to a broad range of industrial applications, from biotechnology, cloud computing, data analytics and big data challenges.

A set of relevant ambitious applications, including HPC for astrophysics, nuclear physics, neural nets and big data, are needed to define the requirements for ExaNeSt architecture and at later stage evaluate the final solution (sub-objective 1 of the project). The design of the platform is tailored to applications: a Hardware-Software Co-design approach that starts from the collection of application requirements is implemented. This Deliverable is the first step of this activity. The applications will provide the network and I/O tracers to drive the research on interconnect and storage (WP3 and WP4). Moreover, the allocation of shared resources such as interconnect links should be optimized for application requirements.

The role of the applications is not only to contribute to the development and tuning of the platform but also on the validation of congestion avoidance and quality-of-service (QoS) capabilities being designed.

The main selection criterion we adopted here is that applications should be mature enough for being ported and re-engineered on the developed ExaScale class supercomputer.

This project will also port and evaluate current-scale HPC applications against the new platform, and where necessary investigate and optimize the HPC support libraries used by these applications.

This deliverable presents a census of some of the more relevant applications in the different Scientific and Industrial fields covered by this project. They represent the state of the art of the HPC applications and tools. Some of these applications will be then ported and adapted to profit from enhancements of the ExaNeSt platform.

1.2 Target Communities

It is worth here to mention the user communities that such census of application involves. Numerous and diverse scientific communities are represented ranging from basic science, like computational astronomy, up to material science and climate science that could have a large societal impact. Beside such scientific communities another set of technical applications involves more industrial players: this is the case of engineering and database applications.

Applications have been selected keeping in mind the existence of an active user community behind the application that could benefit from this action.

The wide range of domains covered by the selected applications complement each other and allow for an adequate representation of relevant, state-of-the-art scientific and business applications

In the following section of this document we will present in details the list of applications selected by partners participating in WP2. For all of them a short description is given and then several specific technical details will be presented.

2. The scientific-technical applications

2.1 Material science: LAMMPS

[Contributed by eXact-lab]

LAMMPS [LAM01] is a classical molecular dynamics code that models an ensemble of particles in a liquid, solid, or gaseous state. It can model atomic, polymeric, biological, metallic, granular, and coarse-grained systems using a variety of force fields and boundary conditions.

LAMMPS is widely adopted in material science community and beyond making it the right choice for the project: it has a large community behind and it was designed from the beginning to be parallel without too many requirements – LAMMPS runs on any parallel machine that compiles C++ and supports the MPI message passing library. This includes distributed or shared-memory parallel machines and Beowulf-style clusters.

From the scientific point of view LAMMPS integrates Newton’s equations of motion for collections of atoms, molecules, or macroscopic particles that interact via short- or long-range forces with a variety of initial and/or boundary conditions. For computational efficiency LAMMPS uses neighbor lists to keep track of nearby particles. The lists are optimized for systems with particles that are repulsive at short distances, so that the local density of particles never becomes too large.

Technical details

The current version of LAMMPS is written in C++. Earlier versions were written in F77 and F90. On parallel machines, LAMMPS uses spatial-decomposition techniques to partition the simulation domain into small 3d sub-domains, one of which is assigned to each processor. Processors communicate and store “ghost” atom information for atoms that border their subdomain. LAMMPS is most efficient (in a parallel sense) for systems whose particles fill a 3d rectangular box with roughly uniform density. Papers with technical details of the algorithms used in LAMMPS are listed in on the web site of the package.

We highlight here some LAMMPS features, of interest for the ExaNeSt project and us:

- runs on a single processor or in parallel
- distributed-memory message-passing parallelism (MPI)
- spatial-decomposition of simulation domain for parallelism
- open-source distribution
- highly portable C++
- optional libraries used: MPI and single-processor FFT
- GPU (CUDA and OpenCL), Intel(R) Xeon Phi(TM) coprocessors, and OpenMP support for many code features
- easy to extend with new features and functionality
- runs from an input script

- syntax for defining and using variables and formulas
- syntax for looping over runs and breaking out of loops
- run one or multiple simulations simultaneously (in parallel) from one script
- build as library, invoke LAMMPS thru library interface or provided Python wrapper
- couple with other codes: LAMMPS calls other code, other code calls LAMMPS, umbrella code calls both

license

LAMMPS is a freely available open-source code, distributed under the terms of the GNU Public License. All versions can be downloaded from the LAMMPS WWW Site. It is distributed by Sandia National Labs.

2.2 Climate science: REGCM

[Contributed by eXact-lab]

Regional climate models (RCMs) are widely used tools to produce high resolution climate simulations at regional scales. The ICTP regional climate modeling system, RegCM, is one of the most used RCMs worldwide, with applications ranging from regional process studies to paleoclimate, climate change, chemistry-climate and biosphere-atmosphere interactions. A new version of the model, RegCM4, has been completed and released in 2012[RCM01]. The current developing version, RegCM 4.5, introduces new features such as the CLM4.5 land surface scheme, a new cloud microphysics scheme and coupling with the MIT Ocean model.

RegCM4 is a regional climate model based on the concept of one-way nesting, in which large scale meteorological fields from a Global Climatic Model (GCM) run provide initial and time-dependent meteorological boundary conditions for high resolution simulations without any active feedback. The RegCM4 is a hydrostatic, compressible, sigma-p vertical coordinate model, running on an Arakawa B-grid n which wind and thermodynamical variables are horizontally staggered. A time-splitting explicit integration scheme is used, in which the two fastest gravity modes are separated from the model solution and then integrated with smaller time steps.

To avoid breaking of the Courant-Friedrichs-Lewy condition, a fixed base time-step is used in the dynamical core which is numerically proportional to the horizontal spatial resolution. As the model is hydrostatic, resolutions lower than 20km are not permitted. The dynamical core computation requirements are thus roughly proportional to the total number of the grid points.

The model uses also multiple physics parameterization to represent subgrid phenomena (surface processes, radiative transfer, planet boundary layer, explicit moisture, optical active aerosols, cumulus convection).

Technical detail

I/O consideration

A complete run of RegCM usually requires the following input set:

- A DOMAIN file to localize the model on a word region. This file contains the localized topology and land-use databases, as well as projection information and land sea mask. This is generally negligible in size for domain under consideration (200x200x25)

- A ICBC (Initial Condition Boundary Condition) set of files that contains the result of a global simulation over the region that RegCM will simulate for all the period of simulation.

In the ICBC files, the overall simulated time should be taken into consideration. In particular there can be from 4 to 6 fields that associate one floating point number to each cell for every time step. Therefore, we have that the dimension of the file is more or less of the order of 6 * 200 * 200 * 25 *4~ 24Mb for each time step. A usual time step is 6 hours long and so for one month of simulation we need one 1GB of data. Considering that a complete, medium-size simulation can cover 150 years bringing the size of ICBC input could easily reach more than 1TB of data in input.

For what concerns the output, it is divided in five files: * ATM * RAD * SAV * SRF * STS. The first two files (ATM and RAD) are roughly as big as the ICBC input file and are the biggest ones. The SAV file, instead, contains a snapshot of the overall execution of the software. It is usually smaller than the other files because it is not time dependent. The SRF and the STS files, instead, are bi-dimensional (they do not contain information that depend on the altitude) and, therefore, are about 25 times smaller than the ATM. Output data can be therefore be more than a factor of two of input data.

RegCM technical details

RegCM performs a 2D Cartesian domain decomposition of the space where the simulation is located. On each step, each process needs to communicate with all its neighbors; for the process that are not on the border of the region, this means 4 communications (two in the North-South direction and two in the East-West direction). If certain climatic conditions are triggered, it is also necessary to transfer the value on the corners of the diagonal cells. This creates really small communications (one floating point number) among the cells in the diagonal directions. When the output must be written on the disk (once every several time steps), one process collects all data from the other processes and write it on the disk. For this reason, there are some massive communications from all the processes to the process zero, which is in charge of save the current status of the simulation. A detailed benchmarks analysis has been conducted in [RCM02]

License

RegCM4 has been made public under GPL license by the authors in 2010 and is available as tar.gz file on eforge web site hosted at ICTP.

Main authors are those listed in the official publication of the code. Graziano Giuliani from ICTP is the actual maintainer and main developer.

Interest for the scientific community

RegCM4 is routinely used by a large number of scientific teams from all over the world in the area of regional climate simulations. Downloads are in the order of several thousands from the 2010 on and more than 1000 people are currently enrolled in official mailing list. Many publications have been achieved by means of the code and the official publication which described the 4.0 version received in less than three years more than 250 citations.

2.3 High Energy Physics

[Contributed by INFN]

2.3.1 LQCD: Lattice Quantum Chromo-Dynamic

QCD (Quantum Chromo-Dynamics) is the field theory describing the physics of the strong force ruling the behavior of quarks and gluons. In general, the solution of the theory is not possible through pure analytical methods and requires simulation on computers. LQCD (Lattice-QCD, for a general introduction see [LQCD01]) is the discretized version of the theory and it is solved using MonteCarlo methods on powerful computers. The related algorithms are considered as one of the most demanding numerical applications.

Present simulations run on 4-D (64^3*128) physical lattices, and are well mapped on a 3D mesh of processors. On a 10+year scale theoretical physics will need to grow to 128^3*128 physical lattices. Since the required computational power scales with the seventh power of the lattice size, n*PETAFlops systems are needed.

LQCD has been the driving force for the design of supercomputers either custom and dedicated, like QCDOC (Columbia University) and several generations of systems designed by INFN (APE family of massive parallel computers), or more general purpose systems like IBM Blue-Gene. Heterogenous PC cluster systems, mainly based on GPU accelerators, have been proved very effective in LQCD codes execution (QuOnG, Fermi,...).

Algorithmic characteristics and technical details

The elementary computation of Lattice QCD computation is the application of the “Dirac operator” to the field φ_y :

$$M_{xy}[U]\varphi_y = \{\delta_{xy} - \kappa \sum_{\mu} [(1-\gamma^{\mu}) U_{\mu,x} \delta_{x+\hat{\mu},y} + (1+\gamma^{\mu}) U^{+}_{\mu,x} \delta_{x-\hat{\mu},y}]\}\varphi_y = \{\delta_{xy} - kD_{xy}[U]\}\varphi_y$$

where x and y span the full 4-dim space-time lattice and $\hat{\mu}$ is the versor connecting to the neighbors in the lattice.

A typical LQCD code repeats the same computation for each point of a large 4-dim discretized space- time, with periodic boundary conditions. The elementary data type of Lattice QCD is the complex number (two floating-point values to represent the real and imaginary part). The computational kernel of Lattice QCD is based on matrix multiplications, where each element of the matrix is a complex number. The computation needed to update the values associated to each point of the discretized lattice, requires to access only the *local data* values of data structures associated to the point itself and its 8 first neighbours remote data along the *X*+, *X*-, *Y*+, *Y*-, *Z*+, *Z*-, *T*+, *T*- axes. Using the so called “frame based computational method”, we can separate (and overlap) the computing from the communication phase gained a raise in the execution efficiency (potentially a factor 2). Those application characteristics, *homogeneity* and *locality*, clearly indicate that a parallel machine with N-dimensional, first-neighbours, toroidal physical interconnection are optimally suited for execution of such computation.

In summary for a typical LQCD code on a 64^3*128 lattice size, we can list few critical application requirements:

- **~10^{22} flops** of sustained aggregated computing power. 300 floating point multiply-accumulate operations (MAC) on complex data is required to compute $M_{xy}[U]\varphi_y$ on a single lattice point. So each inversion of the Dirac Operator on a "small" 24^3x48 lattice requires 2500 applications of the Dirac operator. Actually, one step of integration of the equation of motion requires computing 12/60 times the inverse of the Dirac Operator. The objective of the computation is to generate a set of uncorrelated configuration large enough to get a representative statistical sample to measure the physical observables of interest. A typical set is composed of 1000 uncorrelated configurations.
- **5.48^{10} CW** (complex words), as the minimal required memory size to accommodate the whole complex data set of a 64^3*128 lattice.
- **R=1 CW/T_{MAC}** where **T_{MAC}** is the execution time of a single complex MAC instruction. For a LQCD well-balanced system is the number of required memory bandwidth, measured as a function of the computing node floating point performance.
- **$\rho \leq 10$** where ρ is the ratio between local data access bandwidth and remote data access bandwidth. In other words it measures the required network bandwidth for access remote data of the computing node sub-lattice as a function of local memory data performance.

Available distributions

Several international research collaborations have delivered open source distributions of packages for LQCD simulations, freely available for use in the framework of ExaNeSt project. Most of them are coded in C and C++ and parallel version lever on standard MPI distribution. Examples of these are:

1) USQCD is a collaboration of US scientists developing and using large-scale computers for calculations in lattice quantum chromodynamics. USQCD has developed a suite of software enabling lattice QCD computations to be performed with high performance across a variety of architectures, including both custom facilities and commodity clusters [LQCD02]. This software is made up of highly layered software library modules that can be re-used by higher level applications. Main module is the Chroma package supporting data-parallel programming constructs for lattice field theory and in particular LQCD. It uses the QDP++ data-parallel programming (in C++) and it uses a C++ highly optimized code ported on many architectural systems (single/multiple node workstations and parallelized on multi and many-core clusters of nodes via QMP). Example of lower level library is QUDA [LQCD03] that is a library for performing calculations in lattice QCD on graphics processing units (GPUs), leveraging NVIDIA's CUDA platform. Use of many GPUs in parallel is supported throughout, with communication handled by QMP or MPI.

2) "openQCD" [LQCD04] is an open source, highly optimized reference code for LQCD simulations available under the terms of the GNU GPL License. The package is based on the Hybrid MonteCarlo (HMC) algorithm and several advanced techniques are implemented. The code can be configured at run time through a structured input parameter file. According to openQCD website, "...the programs parallelize in 0,1,2,3 or 4 dimensions, depending on what is specified at compilation time. They are highly optimized for Intel or AMD processors, but will run correctly on any system that complies with the ISO C89 (formerly ANSI C) and the MPI 1.2 standards. For the purpose of testing and code development, the programs can also be run on a desktop or laptop computer. The only requirement is a compliant C compiler and a local MPI installation such as OpenMPI."

2.4 Astrophysics

[Contributed by INAF]

2.4.1 GADGET

GADGET (GAlaxies with Dark matter and Gas intEracT) is a scientific code aimed to solving the gravitational and hydrodynamical equations that rules the formation and evolution of cosmic structures. 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 in "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 modeled 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 parallelized using a hybrid model, MPI+OpenMP. Currently, the OpenMP parallelization is only useful for a subset of interesting physical cases. 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.

Gadget needs the following libraries: MPI, FFTW, GSL, HDF5. I/O is serial (no MPI I/O is performed) and user-configurable. Check pointing and restarting are implemented.

License

GADGET-1 has been made public by the authors in 2001 [GAD01] A subsequent public version has been released in 2005 ([GAD02]). GADGET-2 is distributed under GPL license. A developer version that contains a large number of technical and physics improvement, called GADGET-3, is private. INAT-OATs has access to Gadget-3 and is in the development team. GADGET-3 is available upon request to the main authors. A new GPL public version, GADGET-4, is expected in Spring 2016.

The main authors are Volker Springel, Naoki Yoshida, Simon White. In the course of time, strong contribution came from several research groups in Europe, including (but not limited to) that leaded by K. Dolag in Munich and our, leaded by S. Borgani, in Trieste. Locally, our group has a particular expertise in the implementation of astrophysics modules and in the SPH sector.

The public version can be obtained at the site http://wwwmpa.mpa-garching.mpg.de/gadget/

Interest for the scientific community

GADGET-3 is routinely used by a number of scientific teams (especially European ones). A number of important scientific results have been obtained using this code: cosmological simulation of the formation of individual disk galaxies ([GAD03]) and simulation of formation and evolution of galaxy clusters ([GAD04]). Many different versions of GADGET were used in recent code-comparison papers, like "The Aquila comparison project" ([GAD05]). and "nIFTy galaxy cluster simulation" ([GAD06]). Those two papers give an up-to-date picture of the diffusion of Gadget into the scientific community.

2.4.2 SWIFT

SWIFT [SWI01] is a Tree+SPH code. The numerical techniques it uses are similar to those described above, for GADGET. Note that the gravity is only solved using a Tree code, no PM is present. This may results in a slow-down in the initial phase of cosmological simulations.

The aim of the code is to tackle the challenge of running particle simulations with a very large dynamic range - arising for example in problems of compressible hydrodynamics or galaxy formation - efficiently on modern computer architectures. Such architectures combine many levels of parallelism, using shared memory nodes of many cores, some of which may have additionally an accelerator.

The main bottleneck of such simulations is load imbalance, arising when calculations on a core depend on those performed on another core. Such interdependency severely limits strong scaling behavior. Swift also tackles the issue of how to distribute work if not all cores are equal - as is the case when nodes contain accelerators. Finally, the speed with which cores do work is often limited by the rate at which data gets fed to it: cache-efficiency of the code is crucial.

As it is distributed, SWIFT does not contain any additional physics module: it just performs gravitational and hydrodynamics calculations. It can work in comoving coordinates; it is thus ready for cosmological computations, but a state-of-the-art simulation would require to implement from scratch in this code the needed astrophysical processes described above (see the section GADGET).

Technical details

Task-based parallelism to exploit shared-memory cores as well as on external devices. SIMD vectorization and mixed-precision computation using a gather-scatter paradigm and the use of single-precision values where excessive accuracy is unwarranted. This is supported by the underlying algorithms which attempt to maximize data locality such that vectorization is even possible, and maximises cache throughput.

Hybrid shared/distributed memory parallelism, using the task-based schemes. Parts of the computation are scheduled only once the asynchronous transfers of the required data have completed. Communication latencies are thus hidden by computation, providing for strong scaling across multi-core nodes. Graph-based domain decomposition, which uses information from the task graph to decompose the simulation domain such that the work, as opposed to just the data, as in other space-filling curve schemes, is equally distributed amongst all nodes parallelism. This provides fine-grained load balancing enabling strong scaling, combined with mixing communication and computation, both on each nod.

SWIFT is written in C and implements its own scheduler for implementing its task-based parallelism. The code needs the following libraries: MPI, HDF5. Optionally it requires METIS, which is used to optimize the load between MPI tasks.

License

The code is public under GPL license and can be obtained at the following site:

http://icc.dur.ac.uk/swift/

The main authors are Pedro Gonnet, Matthieu Schaller, Aidan Chalk, Peter W. Draper, Bert Vandenbrouck.

Interest for the scientific community

SWIFT is a new code, gravity was added only in 2013 ([SWI01], [SWI02]). It is being developed by the Durham cosmology group, which is very active in its reference scientific community. Although the code has potential, at the moment no scientific usage has been presented.

2.4.3 CHANGA

Similarly to GADGET and SWIFT, also CHANGA is a Tree+SPH nbody code. It performs collisionless N-body simulations. It can perform cosmological simulations with periodic boundary conditions in commoving coordinates or simulations of isolated stellar systems. It also can include hydrodynamics using the Smooth Particle Hydrodynamics (SPH) technique. It uses a Barnes-Hut tree to calculate gravity, with hexadecapole expansion of nodes and Ewald summation for periodic forces. Timestepping is done with a leapfrog integrator with individual timesteps for each particle.

CHANGA inherited a number of characteristics from GASOLINE ([CHA01]), that is not public. This is due to the fact that the two codes share a number of developers. In particular, the code includes a number of astrophysics module as e.g. gas cooling, star formation, energy feedback from SNII, but not AGB feedback nor stellar evolution.

CHANGA's novel feature is to use the dynamic load balancing scheme of the Charm++ runtime system in order to obtain good performance on massively parallel systems.

Technical details

CHANGA is written in C++. It requires the CHARM++ libraries, used both for the communications and for the task distribution/load balance. It does not require other libraries. CHARM is included in the software distribution package. We note that MPI libraries are not required.

As happened with SWIFT, the task-based parallelism allows this code to scale reasonably well (on specific computers) when a large number of cores is used (see [CHA02]).

This code also features some support for CUDA and SMP.

License

The code is distributed at the following site:

http://www-hpcc.astro.washington.edu/tools/changa.html

under the GPL license. CHARM++ has a slightly different non-commercial, non exlusive user license.

The main authors are Graeme Lufkin, Tom Quinn, Rok Roskar, Filippo Gioachin, Sayantan Chakravorty, Amit Sharma, Pritish Jetley, Lukasz Wesolowski, Harshitha Menon, Edgar Solomonik, Celso Mendes, Joachim Stadel, and James Wadsley.

Interest for the scientific community

Also CHANGA is a quite new code and has not yet been used for scientific usage. However, it has been developed by the Astrophysics group of the Washinghton University (Seattle, U.S.A.), lead by Tom Quinn. The same group wrote the code GASOLINE ([CHA01]), from which CHANGA inherits the majority of its computational techniques. GASOLINE is especially used in the U.S.A. scientific community, but there are also a number of European groups that are allowed to use it. GASOLINE has been successfully used especially in the context of disk galaxy formation (see e.g. [CHA03],[CHA04],[CHA05]).

2.4.4 PINOCCHIO

In currently favored, dark matter dominated cosmological models, initially small density fluctuations are amplified by gravity and eventually condense to form gravitationally-bound systems the so called dark matter halos. The properties of the halo population are of fundamental importance for understanding galaxy and galaxy cluster formation and evolution.

The formation of dark matter haloes can be studied using numerical simulations which usually evolve a set of equal mass particles that represent the dark matter in a periodic simulation box (e.g. GADGET, SWIFT, CHANGA). Also semi-analytical methods can be used to study the formation and evolution of haloes. Semi-analytical techniques have the great advantage to be much faster than the N-Body simulations however they are an approximation to the full non-linear gravitational problem of hierarchical structure formation in a cosmological setting. They are commonly able to reproduce accurately the linear phase of the formation but they fail when they enter highly non linear regimes where they can only approximate the dynamical processes.

The PINpointing Orbit-Crossing Collapsed Hierarchical Object Code, is a fast and perturbative approach for generating catalogue of haloes. Given a set of initial conditions (the same of an N-Body simulation) PINOCCHIO produces masses, positions, velocities for a catalogue of haloes. When compared with an N-Body simulation, PINOCCHIO is able to reproduce the evolution of haloes with an accuracy of about 10%.

Technical details

PINIOCCHIO is a C code based on the Lagrangian Perturbation Theory. It consists of two steps. The first step is the estimation of collapse time, it identifies orbit-crossing (OC) as the instant at which a mass element undergoes collapse. It computes OC numerically by applying local ellipsoidal collapse approximation to the full Lagrangian perturbative expansion. This requires to solve in parallel some differential equations that can be done using fast Fourier transform. PINOCCHIO rely on FFTW v3 and MPI library to calculate OC.

At this stage PINOCCHIO makes no prediction of the mass of the collapsed halo that the particle accreted onto. In fact, the collapsed mass element will not necessarily have accreted onto any halo, but may instead have become part of a filament or sheet since these have undergone OC as well. These structures trace the moderate over-densities that connect the much higher density collapsed halos in simulations.

The Second part of the code assembles mass particles into collapsed halos and OC filaments. The grouping of OC particles into halos mimics the hierarchical formation of objects, and also the way in which halo finders identify collapsed objects in N-Body simulations. Mass particles must be redistributed along the processors using MPI while OpenMP is used to calculate the fragmentation in a shared memory environment.

PINOCCHIO needs the following libraries: MPI, OpenMP, FFTW3, GSL. I/O is serial and user-configurable.

License

The code is public under GPL license and can be obtained at the following site:

http://adlibitum.oats.inaf.it/monaco/Pinocchio/index.html

and it is developed by P. Monaco, T. Theuns and G. Taffoni.

Interest for the scientific community

PINOCCHIO is used by the astrophysical community to generate catalogue of haloes to use to study galaxy and cluster formation problems (e.g. MORGANA Code).

It is extremely useful when large catalogue of haloes must be simulated in short time and it will be used by Euclid space mission to generate mock catalogues of haloes.

2.5 Engineering

[Contributed by EnginSoft]

In the Engineering field two applications related to CFD have been selected.

2.5.1 CFD Application: OpenFOAM

OpenFOAM is a general-purpose open-source CFD code, it is written in C++ and uses an object-oriented approach, which makes it easy to extend. The package includes modules for a wide range of applications.

Its ancestor, FOAM, was written by Henry Weller and others at Imperial College. For a few years, the company Nabla Ltd. sold FOAM as a commercial code. However, in 2004 FOAM was released under GPL and was renamed to OpenFOAM. OpenCFD Ltd. currently distributes OpenFOAM.

Taken from a CFD point of view OpenFOAM is a very capable code, the numerical methods implemented uses the Finite Volume Method on unstructured meshes. It provides many capabilities, including free-surface and multi-phase flow modelling, lagrangian spray model and automatic mesh motion.

For a scientist the code can be considered primarily as a C++ library, used to create executables, known as applications. The applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation. The distribution contains numerous solvers and utilities covering a wide range of problems, as described in this section. One of the strengths of the code is that new solvers and utilities can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved.

Technical details

A central theme of the OpenFOAM design is that the solver applications, written using the OpenFOAM classes, have a syntax that closely resembles the partial differential equations being solved. For example the equation

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot \phi \mathbf{U} - \nabla \cdot \mu \nabla \mathbf{U} = -\nabla p$$

is represented by the code:

```
solve
(
   fvm::ddt(rho, U)
 + fvm::div(phi, U)
 - fvm::laplacian(mu, U)
   ==
 - fvc::grad(p)
);
```

This and other requirements demand that the principal programming language of OpenFOAM has object-oriented features such as inheritance, template classes, virtual functions and operator overloading. These features are not available in many languages that purport to be object-orientated but actually have very limited object-orientated capability, such as FORTRAN-90. C++, however, possesses all these features while having the additional advantage that it is widely used with a standard specification so that reliable compilers are available that produce efficient executables. It is therefore the primary language of OpenFOAM.

A list with technical details of the solvers available can be found in the on line manual. Some generic code capabilities, intended as ExaNeSt project interest follows:

- runs on a single processor or in parallel
- distributed-memory message-passing parallelism (MPI)
- spatial-decomposition of simulation domain for parallelism
- open-source distribution
- highly portable C++
- parallel execution uses the public domain OpenMPI implementation of the standard message passing interface (MPI)
- easy to extend with new features and functionality
- runs from an input script

license

OpenFOAM is a freely available open-source code, distributed under the terms of the GNU Public License. All versions can be downloaded from the openFOAM Site.

2.5.2 CFD Application: SailFish CFD

Sailfish is a free computational fluid dynamics solver based on the Lattice Boltzmann method and optimized for modern multi-core systems, especially GPUs (Graphics Processing Units).

The solver is based on the Lattice Boltzmann Method, which is conceptually quite simple to understand and which scales very well with increasing computational resources.

The Sailfish project is also an experiment in scientific computing and software engineering. Unlike the majority of CFD packages, which are written in compiled languages such as C++ or Fortran, Sailfish is implemented in Python and CUDA C/OpenCL.

This combination is really powerful one, making it possible to significantly shorten development time without sacrificing any computational performance.

The general goals of the project are as follows:

- Scalability: the code is designed to scale well with increasing number of compute cores.
- Agility and extensibility: by implementing large parts of the code in a very expressive language (Python), aiming to encourage rapid experimentation. Run-

ning tests, playing with new boundary conditions or new models is easy, and it often only requires changing a few lines of the kernel code.

- Maintainability: the code is clean and easy to understand. The Mako template engine makes it possible to dynamically generate optimized code
- Ease of use: defining new simulations and exploring simulation results is simple and many details are automated and by default hidden from the end-user.
-

Technical details

The lattice Boltzmann equation (LBE) is a minimal form of Boltzmann kinetic equation which is meant to simulate the dynamic behaviour of fluid flows without directly solving the equations of continuum fluid mechanics. Instead, macroscopic fluid dynamics emerges from the underlying dynamics of a fictitious ensemble of particles, whose motion and interactions are confined to a regular space-time lattice. Technically, the distinctive feature of LBE is a dramatic reduction of the degrees of freedom associated with the velocity space. In fact, particle velocities are restricted to a handful of discrete values $\vec{v} = \vec{c}_i, i=0,b$, by assuming that at each site the particles can only move along a finite number of directions. By endowing this set with sufficient symmetries to fulfill the basic conservation laws of mass, momentum and energy, the LBE can be shown to quantitatively reproduce the equations of motion of continuum fluid mechanics, in the limit of long wavelengths as compared to the lattice scale.

The distinctive features of LBE as a computational solver for fluid problems are its space-time locality, and the fact that information travels along the straight lines defined by the (constant) particle velocities associated with the lattice, rather than along the space-time dependent material lines defined by the flow speed. Due to these properties, the LB approach counts today an impressive array of applications across virtually all fields of fluid dynamics and allied disciplines, such as biology and material science.

Some generic code capabilities, intended as ExaNeSt project interest follows:

- runs on a single processor or in parallel on pure CPU nodes or CPU+GPU nodes
- open-source distribution
- developed in python, with binding for OpenCL & CUDA for parallel paradigm
- easy to extend with new features and functionality
- runs from an input script

license

SailFISH is licensed under the LGPL v3. The project documentation is licensed under the Creative Commons Attribution-ShareAlike 3.0 license. All versions can be downloaded from the SailFISH gitHUB repository.

2.6 Neuroscience: Distributed simulation of polychronous and plastic spiking neural networks

[Contributed by INFN]

A natively distributed mini-application benchmark representative of plastic spiking neural network simulators (DPSNN-STDP) has been developed by INFN, (see [DSP01], [DSP02], [DSP03]). Starting from 2016, it will be used in the framework of the Human Brain Project WaveScalES experiment, coordinated by INFN. The architecture of DPSNN is largely inspired by the large scale neuro-synaptic simulators developed by Izhikevich, and Edelman (2008)., Modha et al. (2011) and Furber et al. (2012). Processes describe synapses in input to cluster of neurons with an irregular interconnection topology, with complex inter-process traffic patterns broadly varying in time and per process. It can be used to gauge performances of existing computing platforms and drive development of dedicated future parallel/distributed computing systems.

The application, designed to be natively distributed and parallel, is a mixed time and event driven spiking neural network simulator, coded as a network of C++ processes equipped with a message passing interface, compatible with standard GNU/Linux plus MPI. Each C++ process describes and simulates a cluster of neurons and their incoming synapses. At each simulation step, the processes exchange among them information about spikes happened in the previous simulation step. The messages travelling between processes are sets of “axonal spikes”, carrying info about the identity of neurons that spiked and the original emission time of each spike. We measured code profiling and scaling figures up to 1024 software processes on a server platform (dual-socket, eight-core Intel Xeon Haswell E5-2630v3@2.4GHz nodes), for a neural network configuration based on columns of Leaky Integrate and Fire neurons with spike-frequency adaptation due to calcium- and sodium-dependent after-hyperpolarization currents. We demonstrated the ability to simulate a grid of 96x96 neural columns, containing a total of 11.4M neurons and 20.4G equivalent synapses. Assuming a cortical column spacing of about 100 µm, the problem represents a cortical tissue area of about one square centimeter that DPSNN is able to simulate about eleven times slower than real-time on a 1024-core execution platform, with a memory occupation below 34.4 byte-synapse. We also run the DPSNN code on an embedded platform (based on NVIDIA Jetson TK1 boards, see [DSP04]). Comparing the results with the run of the same code on a server platform (dual-socket, quad-core Intel Xeon CPUs E5620@2.4GHz nodes), we observed that we just spent 2.2 micro-Joule per simulated synaptic event on the “embedded platform”, approx. 4.4 times lower than the “server platform”, with a 14.4 times better instantaneous power consumption. On the other side, the server platform is a factor 3.3 faster.

2.6.1 Inter-process communication and MPI primitives

During an initialization phase, the connections between pairs of processes are established according to the synaptic matrix interconnecting the cluster of neurons of the network. This phase carry on in two steps, as detailed in [7]. In the first step, each source process informs other processes about the existence of incoming axons and about the number of incoming synapses to be established. A single word, the synapse counter, is communicated among pairs of processes. Under MPI, this can be achieved by an MPI_Alltoall(). Performed once, and with a single word payload, the negligible cost of this first step creates a cumulative network load proportional to the square of the number of processes. The second step transfers the identities of synapses to be created on each target process. Under MPI, the payload, a list of synapses specific for each pair in the subset of processes to be connected, can be transferred using a call to the MPI_alltoallv() library function. The cumulative load created by this second step is proportional to the product between the total number of processes and the subset of target processes reached by each source process.

After initialization, the simulator enters the iterative simulation phase. At each iteration, spikes are exchanged between pairs of process connected by the synaptic matrix. The delivery of spiking messages can be split in two steps, with communications directed toward subsets of decreasing sizes. During the first step, single word messages (spike counters) are sent to the subset of potentially connected target processes. On each pair of source-target process subset, the individual spike counter informs about the actual payload (i.e. axonal spikes) that will have to be delivered, or about the absence of spikes to be transmitted between the pair. The knowledge of the subset has been created during the first step of the initialization phase. The second step, using spike info, establishes a communication channel only between pairs of processes that actually need to transfer an axonal spikes payload during the current simulation time iteration. Under MPI, both steps are implemented using calls to the MPI_Alltoallv() library function.

2.6.2 Traffic Traces: Statistics / Synthetic Generation: a 3D trace matrix

A 3D matrix containing the traffic generated by a specific configuration of a spiking neural network can be produced executing the DPSNN code, properly configured, on a commodity server cluster. The value stored in each element of the matrix expresses the size of the message to be transferred between two processes at a specific numerical simulation time step. A zero indicates that no communication is required between that pair of processes at that time step. The matrix indexes have the following meaning: -First index: the number indicating the step in the computed application; -Second index: the number of the sending process; -Third index: the number of the receiving process.

The 3D trace matrix here described have two aims: 1)- Produce design requisites: the trace can be analysed, e.g. using Matlab, to extract all the statistical info about communication requirements (or about MPI primitives, when MPI implementation is used); 2)- Validate design specifications: the trace can be read by a network simulator, which could provide an objective measure of the proposed ExaNeSt interconnect.

The 3D trace matrix can be analysed along the three directions, in order to obtain specific info about the communication dynamics of the network. For example, the analysis along the time step index produces a 2D matrix, whose indexes are the identifiers of sender - receiver processes. Sum, max, mean, variance and every other statistical value can be extracted, describing the communication between each pair of processes. If an element in this matrix is zero, this means that those processes never communicated during the whole application. Moreover, the 2D matrix produced, for instance summing the values along the time, could return information about the communication of the MPI_ALLTOALL performed in the first communication step of the initialization phase and permits to restrict, by a great amount, the size of the following multicasts implemented by pairs of MPI_ALLTOALLV issued by each process. Fixing a sender receiver column, and looking at the elements in the fixed column, a histogram can be produced, containing all the required statistics about dynamics (size of packets): mean, mode, median, variance, skewness and so on. Summing along a fixed sender processor, you can get statistics about the bandwidth in output. Fixing a sender receiver column and looking at the product of the matrix elements at different time distance, allows collecting statistics about time correlations in the communication.

The one described above are only few examples of all the possible investigation allowed by the 3D traffic matrix. Several additional analyses can be done on the whole matrix, or on a representative subset of elements, to extract useful information at different hierarchical levels (core, multicore, board, chassis system…), according to the project needs.

3. Databases: MONETDB, An In-Memory Database Application for Big Data Analytics

[Contributed by MonetDB]

MonetDB is an open-source column-based database management system (DBMS) for a broad spectrum of high-performance analytical applications, e.g., in data mining, business intelligence, OLAP, scientific databases, XML Query, text and multimedia retrieval, RDF/SPARQL, GIS applications. MonetDB has been developed at the Dutch national research institute of mathematics and computer science since 1993 [MKB09]. Since 2013, the MonetDB core team has been extended with the developers of MonetDB Solutions, the supporting company of MonetDB. Today, the MonetDB users and developers community spans 130+ countries world-wide in six continents.

MonetDB is primarily designed for data warehouse applications. These applications are characterised by their use of large databases, which are crucial to provide business intelligence and/or decision support. Similarly, in various disciplines of data-driven e-science, a deluge of observational data is collected into a warehouse for subsequent scientific analysis. Therefore, MonetDB is also a good candidate to provide data management solutions for scientific applications.

The design of MonetDB is built around the concept of ***bulk processing*** : simple operations applied to large volumes of data to make efficient use of the hardware for large-scale data processing. This focus on bulk processing is reflected at all levels of the architecture and the functionality offered to the user. MonetDB/SQL provides the full-fledged SQL interface, and guarantees the ACID properties for transactions using Snapshots Isolation. MonetDB's focus on bulk processing is reflected in its efficient support for bulk updates, e.g., bulk (binary) data loading, while individual updates carry a relatively high transactional overhead.

MonetDB often achieves a significant speed improvement over other open-source systems for analytical applications, because of its innovations at all layers of a DBMS. First of all, MonetDB adopts a vertical-fragmentation based storage model (a.k.a.***column store***), which is of particular importance for analytical applications. Other advanced database technologies, such as a modern CPU-tuned query execution architecture, automatic and self-tuning indexes, run-time query optimisation techniques, and a modular software architecture, further contribute to MonetDB's outstanding performance for its target applications.

license

MonetDB is a freely available open-source DBMS, currently distributed under the terms of the Mozillar Public License, version 2.0. See here for more information about the MonetDB license. Various versions of MonetDB can be downloaded from the monetdb.org site.

3.1 Technical Features

3.1.1 Software Stack

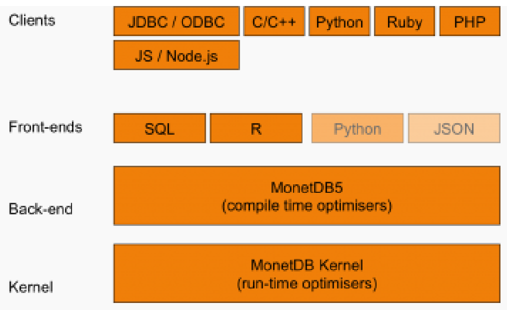

Figure 1: MonetDB software stack

Figure 1 shows the MonetDB software stack. From a user's point of view, MonetDB is a full-fledged Relational DBMS (RDBMS) that supports the SQL:2003 standard. It provides standard client interfaces such as ODBC and JDBC, as well as integration APIs for various programming languages, including C, Java, R, Python, Ruby, Perl, and PHP.

MonetDB is designed to exploit the large main memories and multicore processors of modern computer systems effectively and efficiently during query processing, while the database is persistently stored on disk. With respect to performance, MonetDB mainly focuses on analytical and scientific workloads that are read-dominated and where updates mostly consist of appending new data to the database in large chucks at a time. However, MonetDB also provides complete support for transactions in compliance with the SQL:2003 standard.

Internally, the design, architecture and implementation of MonetDB reconsiders all aspects and components of classical database architecture and technology to achieve the aforementioned performance benefits by effectively exploiting the potentials of modern hardware and enabling extensibility to support new application requirements.

3.2 Challenges to the ExaNeSt Platform

The challenges imposed by a DBMS application like MonetDB on the ExaNeSt platform include i) *performance*, i.e., how can the abundantly available hardware in the system be maximally utilised, and ii) *elasticity*, i.e., to what extend does the platform support scalability in different directions, so as to capture the sudden changes in the application workloads, while minimising energy consumption.

MonetDB performs best when both the hot data set (i.e., data from the database that are required by the running queries) and all intermediates (i.e., data generated by MonetDB during a query execution) fit into the main memory. Therefore, the best practice of getting optimal performance out of MonetDB is to i) keep I/O at an absolute minimal, and ii) have as large as possible I/O bandwidth. Ideally, one should immediately load the whole database into the main memory at the start of a MonetDB server session, and provide sufficient main memory throughout the whole session for the intermediates. In this way, MonetDB would not need any I/O's during the session, unless there are updates. Since updates usually come in batches, large I/O bandwidth and low disk latency enables lower transaction commit time.

The growth of system workloads includes increasing i) data volume, ii) level of concurrency and iii) query complexities. There are basically two ways to cope with workload growth: scale up vertically, or scale out horizontally. The scalability of MonetDB goes in both directions.

3.2.1 Scale Up

One can use a single MonetDB database server to serve an entire data warehouse. The growth of system workloads can be captured to a large extent by plugging in more hardware (CPU, memory, disk, etc). In this setting, there is no inter-node communication. On a single node, important factors include: i) computation power; ii) speed and amount of main memory; and iii) I/O bandwidth and latency.

3.2.2 Scale Out

As of 2015, MonetDB also support distributed query processing, by means of "merge table" [2], "remote table" [3] and "transaction replication" [4]. MonetDB adopts a straightforward master-worker architecture. The master is responsible of dividing a query into subqueries to be sent to worker servers for execution, and conducting the final aggregation of subquery results to produce end results. Depending on the data partitioning scheme, the master might host some parts of the data itself. In this setting, there will be inter-nodes communication between the master and the workers. At the moment, the workers do not communicate with each other. The communication rate is low: one round trip between the master and the workers is triggered by each distributed query. A round trip consists of sending subquery plans from the master to each worker, and subquery results from the workers to the master. The communication volumes (i.e., data exchanged between master and workers) are mostly small (mostly in kilobytes or megabytes, gigabytes data transfer in this case is expected to be rare), since most queries are read-only queries that return aggregated data. Only bulk updates queries, database repartitioning, database migration [5] or database duplication will cause a burst of high communication volumes. In such cases, data transfer, through I/O and/or network, involving 100s of gigabytes to 10s of terabytes are common in current applications. This will be discussed in more details in deliverable D3.1.

3.2.3 Elasticity

Elasticity is not only about the ability of keeping up performance under increasing workloads, which is usually done by allocating more system resources for the demanding applications, but also about the ability of reducing resource usage when the workloads go down. A system with high level of elasticity helps reduce both the operational costs and the environmental burden. In the scale in which the ExaNeSt platform will operate, elasticity is of particularly importance.

Based on earlier results of benchmarking MonetDB energy consumption, the best way to minimise energy consumption is to finish a query execution as fast as possible, then immediately shut down as many as possible system resources, such as the hard disk. Hence, to facilitate both *elasticity* and *low energy consumption*, the ExaNeSt platform must be able to quickly detect and turn-off idle system resources, as well as be able to instantly bring a whole database instance (which might go up to terabytes, or even petabytes) back on-line. Once again, this requirement put high pressure on the I/O speed.

4. Global Overview

[Contributed by all partners]

The table below summarizes the key properties of the applications that have surveyed in this census. The list covers a broad spectrum of relevant science and business domains. In this section for each application, presented previously, we identify the programming, essential storage, and communication characteristics, their current and projected scales, as well as their principal bottlenecks in today's HPC machines. We can in this way compare them side by side and provide a synthetic overview of all of them.

Application	Programming model	Storage/ communication characteristics	Scale: current / target	Bottleneck
GADGET[1] (N-Body + Hydrodyn)	MPI OpenMP	100+ TBs/sim; Bandwidth intensive at checkpoint/restart; latency-sensitive compute common; non nearest neighbor in principle is always needed	up to 100B particles / 10-100x better force resolution needed Multi-scale simulations / 10-50x better force resolution needed	Interconnect and storage /data. Memory size (for load balancing) & system, (low-latency) interconnect
SWIFT (N-Body + Hydrodyn)	MPI + Pthreads	100+ TBs/sim; Bandwidth intensive at checkpoint/restart; latency-sensitive compute common; non nearest neighbor in principle is always needed	up to 100B particles / 10-100x better force resolution needed Multi-scale simulations / 10-50x better force resolution needed	Interconnect and storage /data. Memory size (for load balancing) & system, (low-latency) interconnect
CHANGA (N-Body + Hydrodyn)	C++ and Charm++	100+ TBs/sim; Bandwidth intensive at checkpoint/restart; latency-sensitive compute common; non nearest neighbor in principle is always needed	up to 100B particles / 10-100x better force resolution needed Multi-scale simulations / 10-50x better force resolution needed	Interconnect and storage /data. Memory size (for load balancing) & system, (low-latency) interconnect
PINOCCHIO	MPI+ OpenMP	Mainly MPI FFT communications (fftw_MPI_	Several 100s core/several 1000s of cores	Interconnect and amount of memory per core.

[1] The Astrophysical N-Body and Hydrodynamic codes have similar characteristics and similar storage, interconnect and scalability.

(DPSNN-STDP)	C++ MPI	Mainly multicast communications (all-to-all, all-to-many). Runs on commodity clusters & APEnet+.	Several 100s cores / at least 1000x in next five years	Interconnect for higher-resolution and multi-scale/region simulations
Lattice QCD	C (C++) MPI SIMD	Nearest neighbor communication; embarrassingly parallel. Several open source distributions available. Used as benchmark for all classes of HPC systems	Scales as you add nodes.	Compute bottlenecks; number of nodes in system.
OpenFOAM	MPI	100+ GBs/sim; Bandwidth intensive at checkpoint/restart; latency-sensitive compute common; non nearest neighbor as particles move.	up to 100M cells/ 100-1000x needed	Interconnect and storage /data.
LAMMPS	MPI + OpenMP	Nearest neighbor communication	1K-10K atoms/ more than 100x needed	Placing tasks close to data
RegCM	MPI	Data parallel application; from 1 up to 10 TB per simulation; several interacting sims run in parallel.	400x800x20 levels per sim; 2-3 concurrent sims / need 2x	Storage / data bottleneck. Network bottleneck mainly due to latency
MonetDB	C, Column-store DB	Irregular communication, data-intensive, bandwidth and latency-critical	Current 100sGB to 10sTB / Desired 100sTB; more interactive.	Compute, memory size, interconnect (for bulk updates) and storage

5. Conclusions

[Contributed by all partners]

In this document we presented several scientific and technical applications that could be of interest for the ExaNeSt platform. For each of them we briefly presented the scientific/ technical context and then we discussed technical and implementation details that should be taken into account by the project. It is likely that not all of them could be ported and benchmarked on the new platform.

Mainly open source applications have been selected keeping in mind high standards in terms of scientific maturity of the applications and the associated needs toward exascale usage.

6. References

Here below we collect some references related to the specific codes presented previously. The list is organized by scientific code and/or application.

DISTRIBUTED SIMULATION OF POLYCHRONOUS AND PLASTIC SPIKING NEURAL NETWORKS (DPSNN-STDP)

[DSP01] P.S. Paolucci et al., "Dynamic Many-process Applications on Many-tile Embedded Systems and HPC Clusters: the EURETILE programming environment and execution platforms", Journal of Systems Architecture, Available online 24 November 2015, ISSN 1383-7621, http://dx.doi.org/10.1016/j.sysarc.2015.11.008

[DSP02] E. Pastorelli, et al., "Scaling to 1024 software processes and hardware cores of the distributed simulation of a spiking neural network including up to 20G synapses", (2015) arXiv:1511.09325, http://arxiv.org/abs/1511.09325

[DSP03] E. Pastorelli, et al., "Impact of exponential long range and Gaussian short range lateral connectivity on the distributed simulation of neural networks including up to 30 billion synapses", (2015) arXiv:1512.05264, http://arxiv.org/abs/1512.05264

[DPS04] P.S. Paolucci, et al. "Power, Energy and Speed of Embedded and Server Multi-Cores applied to Distributed Simulation of Spiking Neural Networks: ARM in NVIDIA Tegra vs Intel Xeon quad-cores", (2015) http://arxiv.org/abs/arXiv:1505.03015

MONETDB

[BMK99] P. A. Boncz, S. Manegold, and M. L. Kersten. Database Architecture Optimized for the New Bottleneck: Memory Access. In Proceedings of the International Conference on Very Large Data Bases (VLDB), pages 54–65, Edinburgh, Scotland, UK, September 1999. Received 10-year Best Paper Award at VLDB 2009 [MKB09].

[MKB09] S. Manegold, M. L. Kersten, and P. A. Boncz. Database Architecture Evolution: Mammals Flourished long before Dinosaurs became Extinct. Proceedings of the VLDB Endowment (PVLDB), 2(2):1648–1653, August 2009. 10-year Best Paper Award paper for [BMK99].

LAMMPS

[LAM01] S. J. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19 (1995).

REGCM

[RCM01] Giorgi F, Coppola E, Solmon F, Mariotti L and others (2012) RegCM4: model description and preliminary tests over multiple CORDEX domains. Clim. Res. 52:7-29

[RCM02] T,BJimma, M Scarcia, G Giuliani, S Cozzini (2011) A benchmark study of the RegCM4 package for climate simulations appeared in CLCAR2011 conference

OpenFOAM

[OFM01] B. Mikuz et al. OpenFOAM simulations of the Turbulent Flow in a Rod Bundle with Mixing Vanes. Proceedings of the 21st International Conference Nuclear Energy for New Europe, Ljubljana, 2012.

.

SailFISH

[SFI01] M. Januszewski, M. Kostur, Sailfish: A flexible multi-GPU implementation of the lattice Boltzmann method, Computer Physics Communications 185/9 (2014), DOI:10.1016/j.cpc.2014.04.018, Prof. Sauro Succi, C.N.R. Istituto Applicazioni del Calcolo

LQCD

[LQCD01] G.P.Lepage "Lattice QCD for Novices",http://arxiv.org/abs/hep-lat/0506036 and also https://en.wikipedia.org/wiki/Lattice_QCD

[LQCD02] USQCD website: https://usqcd-software.github.io/

[LQCD03] M. Clark and others, "Solving Lattice QCD systems of equations using mixed precision solvers on GPUs", Comput.Phys.Commun.181:1517-1528,2010, http://arxiv.org/abs/0911.3191

[LQCD04] openQCD website: http://luscher.web.cern.ch/luscher/openQCD/

PINOCCHIO

[PIN01] P. Monaco, T. Theuns, G. Taffoni, "The pinocchio algorithm: pinpointing orbit-crossing collapsed hierarchical objects in a linear density field" (2002) Monthly Notices of the Royal Astronomical Society, Volume 331, Issue 3, pp. 587-608

[PIN02] Taffoni, Giuliano; Monaco, Pierluigi; Theuns, Tom "PINOCCHIO and the hierarchical build-up of dark matter haloes", 2002MNRAS.333..623T

CHANGA

[CHA01] J. Wadsley, J. Stadel, T. Quinn, "Gasoline: a flexible, parallel implementation of TreeSPH" (2004) New Astronomy, Volume 9, Issue 2, p. 137-158

[CHA02] H. Menon, L. Wesolowski, G. Zheng, P. Jetley, L. Kale, T. Quinn, F. Governato,"Adaptive techniques for clustered N-body cosmological simulations" (2015) Computational Astrophysics and Cosmology, Volume 2, article id.1

[CHA03] F. Governato, B. Willman, L. Mayer, A. Brooks, G. Stinson, O. Valenzuela, J. Wadsley, T. Quinn, "Forming disc galaxies in ΛCDM simulations" (2007) Monthly Notices of the Royal Astronomical Society, Volume 374, Issue 4, pp. 1479-1494

[CHA04] F. Governato, C. B. Brook, A. Brooks, L. Mayer, B. Willman, P. Jonsson, A.M. Stilp, L. Pope, C. Christensen, J. Wadsley, T. Quinn, "Forming a large disc galaxy from a $z < 1$ major merger" (2009) Monthly Notices of the Royal Astronomical Society, Volume 398, Issue 1, pp. 312-320

[CHA05] F. Governato, A. Zolotov, A. Pontzen, C. Christensen, S.H. Oh, A.M. Brooks, T. Quinn, S. Shen, L. Wadsley, "Cuspy no more: how outflows affect the central dark matter and baryon distribution in Λ cold dark matter galaxies" (2012) Monthly Notices of the Royal Astronomical Society, Volume 422, Issue 2, pp. 1231-1240.

GADGET

[GAD01] V. Springel, N. Yoshida, S.D.M. White, "GADGET: a code for collisionless and gasdynamical cosmological simulations" (2001) New Astronomy, Volume 6, Issue 2, p. 79-117

[GAD02] Springel, "The cosmological simulation code GADGET-2" (2005) Monthly Notices of the Royal Astronomical Society, Volume 364, Issue 4, pp. 1105-1134

[GAD03] G. Murante, P. Monaco, S. Borgani, L. Tornatore, K. Dolag, D. Goz, "Simulating realistic disc galaxies with a novel sub-resolution ISM model" (2015) Monthly Notices of the Royal Astronomical Society, Volume 447, Issue 1, p.178-201

[GAD04] E. Rasia, S. Borgani, G. Murante, S. Planelles, A.M. Beck, V. Biffi, C. Ragone-Figueroa, G.L. Granato, L.K. Steinborn, K. Dolag, "Cool Core Clusters from Cosmological Simulations" (2015) The Astrophysical Journal Letters, Volume 813, Issue 1, article id. L17

[GAD05] C. Scannapieco, and 22 coauthors, "The Aquila comparison project: the effects of feedback and numerical methods on simulations of galaxy formation" (2012) Monthly Notices of the Royal Astronomical Society, Volume 423, Issue 2, pp. 1726-1749

[GAD06] F. Sembolini, and 26 coauthors, "nIFTy galaxy cluster simulations I: dark matter & non-radiative models" (2016), submitted to MNRAS, http://arxiv.org/abs/1503.06065

SWIFT

[SWI01] J. Schaye, and 21 coauthors, "The EAGLE project: simulating the evolution and assembly of galaxies and their environments" (2015) Monthly Notices of the Royal Astronomical Society, Volume 446, Issue 1, p.521-554

[SWI02] T. Theuns, A. Chalk, M. Schaller, P. Gonnet, "SWIFT: task-based hydrodynamics and gravity for cosmological simulations" (2015), Proceedings of the EASC 2015 conference, Edinburgh, UK, April 21-23, 2015, http://arxiv.org/abs/1508.00115