AVBP and YALES2 Portability, Tuning and Scalability on AMD EPYC 7002 Rome Processors

Guillaume Balarac[1], Patrick Bégou[1], Gabriel Staffelbach[2], Vincent Moureau[3], Francesco Gava[3], Ghislain Lartigue[3], and Christelle Piechurski[4]

[1]LEGI
[2]CERFACS, Toulouse, France
[3]CORIA, CNRS UMR6614, Normandie Université, UNIROUEN, INSA of Rouen, France
[4]GENCI

April 2020

Abstract

High-Performance Computing (HPC) has fostered innovation in a wide range of traditional scientific domains for many years and in the last decade, became a state-of-the-art vehicle for high-fidelity simulations of combustion dynamics on which aeronautics and automotive industries rely on to improve their time-to-market solution as well as the quality of their products. The Exascale perspective changes dramatically the picture and drives scientists to implement new methodologies in their future algorithms to unlock large-scale numerical simulations on many-cores computing architectures. Meanwhile, current Computational Fluid Dynamics (CFD) applications AVBP and YALES2, must be optimized and fine-tuned to run efficiently on pre-Exascale nowadays technologies.

1 Introduction

The simulation of turbulent flows and more particularly turbulent flames is highly challenging. Even if the governing equations of such flows, the Navier-Stokes equations, are well known and even if many models exist, the high-fidelity simulation of turbulence and flames is still difficult. Indeed, in these flows, a wide range of scales coexist, and supercomputers, though extremely helpful, are still not powerful enough to enable the resolution of all these scales. In addition, another important aspect to sort out is the complex geometry of the burners and aerodynamic devices encountered in applications.

To tackle these issues and exploit efficiently the current and future supercomputers, the French CFD academic and industrial community participates to the development of two flow solvers: AVBP and YALES2. They have in common the ability to deal with massive unstructured and hybrid grids. While AVBP [10] targets mainly compressible flows, YALES2 [12, 11, 8] is focused on low-Mach number flows. However, to share best practices and continue to propose the best-in-class performances to the community, LEGI and CORIA have organized the third edition of the yearly Extreme Computational Fluid Dynamics (ECFD) Workshop/hackathon at Autrans near Grenoble, France, in January 2020. This 3rd international workshop gathered 50 researchers and engineers to share knowledge and focus on algorithms optimization as well as new methodologies implementation with the ambitious target to address and unlock some of the well foreseen technological Exascale stakes. One of this well-known technological challenges is relative to the end of Moore's law which lead to increase the core density of computing units. While the many cores architecture was initially implemented only on accelerators technology like the Knights Landing (KNL) from Intel® or on GPU from nVIDIA, the many core architecture have been democratizing also on General Purpose Computing Processing Unit for a few years, reaching now 64 cores processors on the new x86 AMD Rome technology thanks to a MCP (Multi-Chips module) implementation. This should become a standard in the next couple of years for mostly all GPP (General Purpose Processor). This new context drives the CFD community to engage strongly on those challenges, prepare itself for changes, define best-practices and share them, at least for the 2 well-known CFD applications AVBP (CERFACS) and YALES2 (CORIA).

2 Overview of the CFD codes

The YALES2 and AVBP codes are widely used in the French academic combustion community as well as in the aeronautical industry for the design of combustion chambers. Each code also has several other fields of application: YALES2 is mostly dedicated to Low-Mach number flows while AVBP, which is based on the full Navier-Stokes equations, is more suitable to study compressible flows and thermo-acoustics instabilities. The following sections aim at giving more insights on each of these two codes.

2.1 YALES2

The YALES2 code [12, 11, 8] is a massively parallel numerical toolbox on which many solvers have been developed since 2009 to address various physical problems: incompressible flows, combustion, two-phase flows, heat transfer, radiative transfer and much more. All these solvers benefit from the underlying high-performance numerical library: highly-optimized linear solvers, automatic mesh refinement, high-order Finite-Volume integration, parallel I/O, etc. These numerical methods are implemented on unstructured meshes which enable to describe easily and efficiently complex geometries. YALES2 is currently used by more than 300 persons both in academic and industrial sectors. The library is composed of approximately 500'000 lines of Object-Oriented Fortran and its parallelism is managed with the MPI approach. While an hybrid OpenMP-MPI version is under development, the results provided in this white paper are only relative to the YALES2 MPI implementation. From a numerical point-of-view, YALES2 relies on a Node-Centered Finite-Volume approach which naturally guarantees the conservation of the transported quantities. It features various numerical methods, and more particularly a 4th-order convection scheme [2] which is particularly well suited for the Large-Eddy Simulation (LES) approach, as it can transport the eddies present in the flow over long distances without damping them artificially.

The central part of the code is the Low-Mach number Navier-Stokes solver for constant density flows which solves the following set of equations:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho}\nabla P + \nabla \cdot \tau \tag{2}$$

To do so, it relies on the classical projection method proposed by [?] to impose the divergence-free constraint on the velocity, augmented with a recycling of the pressure at the previous timestep to reduce the splitting errors. This method is composed of a first-step known as the *prediction*:

$$\frac{\mathbf{u}^\star - \mathbf{u}^n}{\Delta t} + \nabla \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) = -\frac{1}{\rho}\nabla P^{n-\frac{1}{2}} + \nabla \cdot \tau^n \tag{3}$$

followed by a second step known as the *correction*:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^\star}{\Delta t} = -\frac{1}{\rho}\nabla(P^{n+\frac{1}{2}} - P^{n-\frac{1}{2}}) \tag{4}$$

In this last step, the pressure variation is obtained by solving a Poisson equation to enforce the divergence-free constraint on the velocity:

$$\nabla \cdot \left(\frac{1}{\rho}\nabla(P^{n+\frac{1}{2}} - P^{n-\frac{1}{2}})\right) = \frac{\nabla \cdot \mathbf{u}^\star}{\Delta t} \tag{5}$$

The discretization of this Poisson equation on a mesh will then provide a linear system in which the unknowns are the pressure at each node of the mesh. This linear system can thus be extremely large for real-life applications (the standard mesh size for 3D problems is now between 10M and 500M cells) and can only be solved with dedicated methods, fitted for massively parallel environment. Even with tens of years of R&D on linear solvers, solving the Poisson equation in turbulent flows still accounts for most of the computation cost. Moreover, solving this linear system becomes harder and harder when the mesh size and the number of CPU increases. The first reason is linked to the condition number of the linear system which is actually a measure of the ratio between the largest and the smallest scale in the problem. The other issue is related to the need to perform parallel operation to solve the linear system. In YALES2, this system is solved with an Conjugate Gradient iterative solver which involves one Matrix-Vector product and two dot products at each iteration. This implies one p2p (point-to-point) communication and three all2all reductions at each iteration of the CG (an additional reduction is needed to decide if the system is converged). The cost of these parallel operations increases when more and more processors are used to solve the problem. A classical way to improve the performance of the method, is to precondition the system. In YALES2, this preconditioning is based on a deflation algorithm [8], which first solves the system on a coarser grid and then perform aback-projection of this coarse residual on the fine mesh before performing a classical CG step. This coarse mesh is obtained via a double domain decomposition technique: the full domain is first shared among all the available cores and then each core performs another domain decomposition on its own mesh. Both steps are performed sequentially with either the Metis [**?**] or Scotch [**?**] library. The second level of decomposition provides a set of so-called *cell-groups* which contains a tunable number of elements (usually of the order of 1'000). This technique has two main advantages:

- the coarse mesh is built from these cell-groups, which size determine the preconditioning ratio.
- all the data in the code are allocated on these cell-groups: the size of the arrays is thus limited to a few kilo-bytes which is of key importance to fit in the lowest cache level of the cores.

2.2 AVBP

AVBP [10] is a 3D Navier-Stokes solver for compressible reactive two-phase flows developed by CERFACS since 1997. The code relies on an explicit resolution of the equations using the Large-Eddy Simulation approach where the large structures are resolved and small scales are modeled. The LES models available in AVBP are the Dynamic Smagorinsky [5], WALE[4] and SIGMA[13] models. In order to handle such arbitrary grids, the data structure of AVBP employs a cell-vertex finite-volume formalism. The numerical method is based on a Finite-Element low-dissipation Taylor-Galerkin discretization in combination with a linear-preserving artificial viscosity model. AVBP was built for massively parallel computations on unstructured grids. The code represents approximately 500'000 lines of Fortran code with some C elements. Parallelism follows the single program multiple data approach is based on domain decomposition using the Parmetis (`http://glaros.dtc.umn.edu/gkhome/software`) or Ptscotch (`https://gforge.inria.fr/projects/scotch/`) libraries. Large input/output operations are based on the Parallel HDF (`https://www.hdfgroup.org/solutions/hdf5/`) library with the possibility of having a flat (all processes) or hierarchical (one process out of N) I/O pattern depending on the file system contention. On the fly post-processing algorithms allow for data extraction of iso-surfaces and planes to reduce data output. The cell vertex approach uses cache blocking based on groups of cells to reduce cache errors.

3 Presentation of the three tested architectures

3.1 SuperComputing Platforms

AVBP and YALES2 benchmarks's results provided in Section 5 have all been obtained on GENCI (Grand Equipement National du Calcul Intensif) supercomputing resources. GENCI is the French National Agency in charge of providing High-Performance Computing, Artificial Intelligence and data processing capabilities at national and European level to promote the use of intensive computing associated with Artificial Intelligence for the benefit of French academic and industrial open research communities, fostering and boosting researches for large scale scientific challenges. GENCI is funding 3

national supercomputers, each hosted and operated by its partners: Joliot-Curie (TGCC/CEA), Jean Zay (IDRIS/CNRS) and Occigen (CINES/CPU).All benchmarks results introduced in this document have been produced on the Joliot-Curie supercomputer for Intel®Xeon®Skylake and AMD EPYC Rome processors and on the INTI prototype for Marvell®ThunderX2®processor:
Joliot Curie Supercomputer is one of the 3 French supercomputing facilities, hosted in TGCC (Trés Grand Centre de Calcul) and operated by the CEA (Commissariat à l'Energie Atomique). The machine, a Tiers 0 PRACE (Partnership for Advanced Computing in Europe) system, is currently built on top of 4 heterogeneous partitions (Intel®Skylake, Intel®KNL, AMD Rome, nVIDIA®V100 GPUs) capable to sustain both HPC and AI workloads with more than 22 PFLOP/s peak performance. Its computing capabilities is expected to grow by the end of 2020 with an innovative ARM partition in production dedicated to the European and French research community, to pursue the path to Exascale. The AMD and GPU partitions as the future ARM partition have been also funded by the European Union through the PPI4HPC project (Public Procurement of Innovative solutions for High Performance Computing). This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N0 754271. The multi-partitions machine access data thanks to a datacentric storage architecture based on an extremely fast multi-layer Lustre File Systems with the level 1 reaching 500 GB/s to achieve IO data intensive workloads. Joliot-Curie supercomputer's installation started in 2017. Its first phase based on Intel Skylake and Intel KNL processors entered in production mid-2018 with resources embedded in the Atos Bull Sequana X1000 open, modular and highly energy efficient Platform. The second phase based on AMD Rome and nVIDIA GPUS relies on the X2000 Atos BullSequana platform and entered in production in March 2020. As of interest, only the Intel Skylake and AMD Rome partitions will be described below:

- The Intel Skylake partition contains 1 656 Skylake 8168 bi-sockets nodes (79,488 compute cores), each with 192GB memory, all linked together through a Fat-tree topology 4x EDR (4x 25Gbps) InfiniBand Mellanox network. It has a 6.86PF peak performance.
- The AMD Rome partition contains 2292 AMD EPYC Rome 7H12 bi-sockets node (293,376 compute cores) each with 256 GB DDR4 memory. The nodes are linked together through a Dragonfly+ topology based on 4x HDR (4x 50Gbps) InfiniBand Mellanox network at switch level. It has a 12.2 PFLOP/s peak performance.

All AMD benchmarks results introduced in this white paper have been produced thanks to the TGCC/CEA & GENCI which provided 150,000 compute hours on the AMD Rome partition for a duration of around one month.

ARM benchmarks have been run on the GENCI ARM INTI prototype resources, hosted and operated by the CEA. Results were obtained mostly during the 3rd ARM hackathon organized by GENCI's Technology Watch Activity supported by ARM company. This prototype was opened to the user's community at the end of 2019 and its resources are now accessible via e-dari. It is equipped with 30 dual-socket ThunderX2 nodes, each equipped 64 cores ARMv8 running at 2.2GHz and with 256 GB of memory interconnected through a 4X EDR Mellanox Infiniband network. Those nodes are integrated into the Atos BullSequana X1000 platform.

3.2 AMD EPYC 7H12

The AMD EPYC 7002 Series Processors (Rome) is the second generation of the EPYC x86 platform. Its implementation relies on Zen2 core architecture built on a 7 nm process technology and a multi-die architecture to support up to 64 compute cores on a single SoC (System-On-chip), enhanced IO capabilities through 128 lanes PCIe® Gen4 I/O and 8 DDR4 memory channels (and up to 2DIMMs per channel) running at up to 3200MT/s, boosting memory bandwidth up to 204GB/s Peak. The AMD Rome processor supports up to 280W TDP per socket and a CPU clock speed up to 3.4Ghz Turbo boost CPU clock speed.

Each SoC is built on top of up to 8 CCD (Core Complex Die also named Compute Core Die) with a 7nm process technology and 1 centralized IOD (Input/Output Die) built through a 14 nm process technology to improve NUMA effects. This topology is illustrated in Fig. 2. Each CCD is built on top a 2 CCX (Core CompleX) and can host a maximum of 8 physical Zen2 cores. Up to 4 Zen2 cores per CCX are sharing 16MB L3 cache while L1 (32KB data and 32 KB instruction) and L2 (512 KB data) caches are dedicated per Zen2 core. In addition, each core can support SMT, allowing 2 execution threads to execute simultaneously. Each CCD connects to memory, I/O, and each other through the second generation of the AMD Infinity Fabric (Fig. 1), an AMD proprietary link interconnect providing balanced connection between all components on the SoC. All the cores on a single CCD access memory space address through the same memory controller.

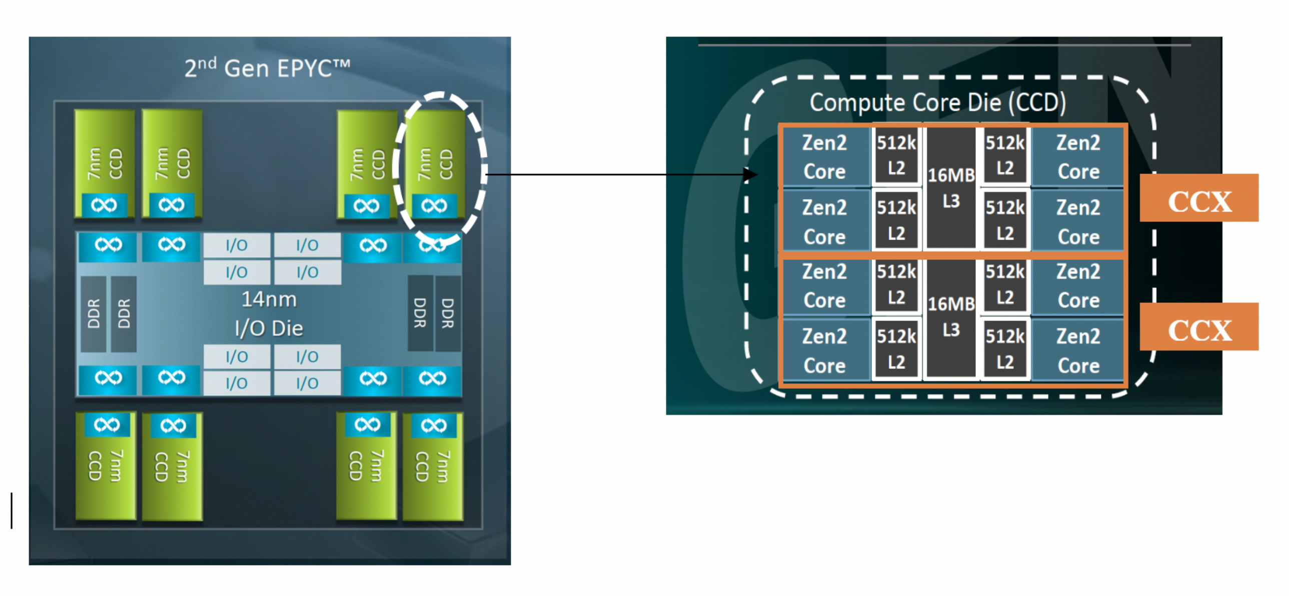


Figure 1: EPYC Rome CCD and CCX Block Diagram

On the compute side, the AMD Rome processor supports AVX2 or so called AVX256 (256 bits width) which means Zen2 core can run AVX-256 instructions (ISA) in one clock. In term of instructions, the Zen2 core is able to support 2x (ADD, FMA) @256 bits, leading to sustain 16 floating-point operations per second per core.

The AMD Rome SKU deployed at large scale on the Joliot-Curie Supercomputer is an off-SKU roadmap called 7H12. The 7H12 SKU has 64 Zen2 cores, each running at 2.6GHz Base clock Speed and up to 3.3 GHz when Turbo boost is enabled, with a maximum of 280W TDP. Each processor is capable of delivering up to 2.66 TFLOP/s DP (Double precision) peak performance (Rpeak). It supports 8 memory channels and up to 2 DIMMs per channel running at 3200 MT/s with 256 MB L3 shared cache and 64 KB (32+32) L1 and 512KB L2 dedicated cache per core. Its micro-architecture is based on a 8 CCD MCM (Micro-Chips Module). Information relative to AMD EPYC 7H12 [1]

The AMD Rome EPYC are available either on single or dual sockets server. For a dual-sockets server, like the ones deployed on Joliot-Curie supercomputer, two sockets are connected through xGMI2 links providing up to 288 GB/s Peak bidirectional bandwidth maintaining 128 lanes PCIe®Gen4 I/O available on the dual sockets platform providing up to 256 GB/s bidirectional bandwidth for I/O capabilities. Each processor is partitioned into four logical quadrants, each being a NUMA domain. There are 4 Numa domains Per Socket (NPS) and up to 8 Numa Domains per node as represented in Fig. 2. Memory is interleaved across the two memory channels in each quadrant. PCIe devices will be local to one of four NUMA domains on the processor depending on the quadrant of the IO die that has the PCIe root for that device.

SPECrate2017_int_base, SPECrate2017_int_peak, SPECrate2017_fp_base, SPECrate2017_fp_peak

[1] https://www.amd.com/fr/products/cpu/amd-epyc-7h12

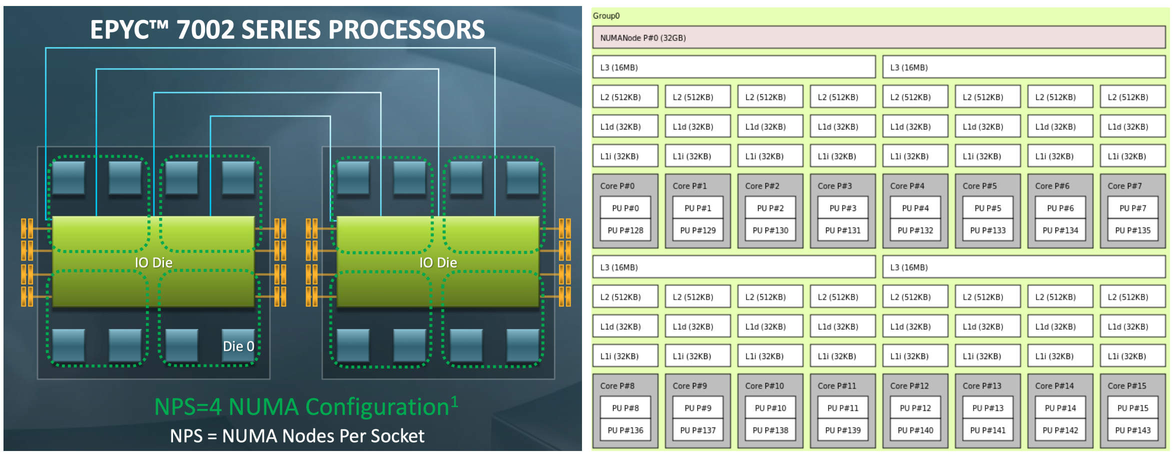
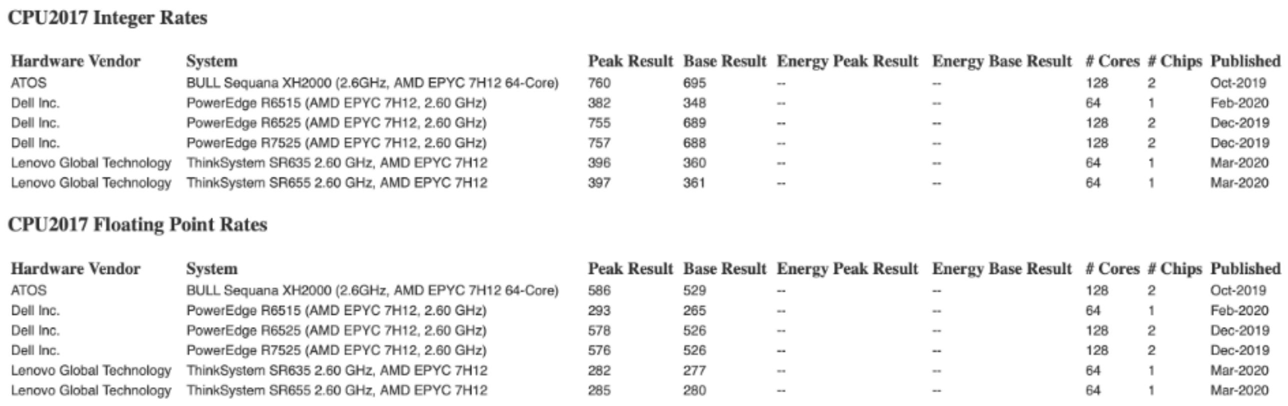


Figure 2: NUMA configuation (left); Core topology (right) for the AMD Epyc Zen2 architecture.

for AMD Rome SKU 7H12 numbers can be found below. They are extracted from spec.org[2] with the highest numbers demonstrated by Atos on the BULL Sequana XH2000 platform deployed at TGCC/CEA on the Joliot Curie supercomputer, thanks to the ability of the platform to cool properly and efficiently the 7H12 AMD SKU.



CPU2017 Integer Rates

Hardware Vendor	System	Peak Result	Base Result	Energy Peak Result	Energy Base Result	# Cores	# Chips	Published
ATOS	BULL Sequana XH2000 (2.6GHz, AMD EPYC 7H12 64-Core)	760	695	--	--	128	2	Oct-2019
Dell Inc.	PowerEdge R6515 (AMD EPYC 7H12, 2.60 GHz)	382	348	--	--	64	1	Feb-2020
Dell Inc.	PowerEdge R6525 (AMD EPYC 7H12, 2.60 GHz)	755	689	--	--	128	2	Dec-2019
Dell Inc.	PowerEdge R7525 (AMD EPYC 7H12, 2.60 GHz)	757	688	--	--	128	2	Dec-2019
Lenovo Global Technology	ThinkSystem SR635 2.60 GHz, AMD EPYC 7H12	396	360	--	--	64	1	Mar-2020
Lenovo Global Technology	ThinkSystem SR655 2.60 GHz, AMD EPYC 7H12	397	361	--	--	64	1	Mar-2020

CPU2017 Floating Point Rates

Hardware Vendor	System	Peak Result	Base Result	Energy Peak Result	Energy Base Result	# Cores	# Chips	Published
ATOS	BULL Sequana XH2000 (2.6GHz, AMD EPYC 7H12 64-Core)	586	529	--	--	128	2	Oct-2019
Dell Inc.	PowerEdge R6515 (AMD EPYC 7H12, 2.60 GHz)	293	265	--	--	64	1	Feb-2020
Dell Inc.	PowerEdge R6525 (AMD EPYC 7H12, 2.60 GHz)	578	526	--	--	128	2	Dec-2019
Dell Inc.	PowerEdge R7525 (AMD EPYC 7H12, 2.60 GHz)	576	526	--	--	128	2	Dec-2019
Lenovo Global Technology	ThinkSystem SR635 2.60 GHz, AMD EPYC 7H12	282	277	--	--	64	1	Mar-2020
Lenovo Global Technology	ThinkSystem SR655 2.60 GHz, AMD EPYC 7H12	285	280	--	--	64	1	Mar-2020

Figure 3: SPECrate2017int and SPECrate2017fp AMD 7H12.

3.3 Intel Skylake 8168

The Intel®Xeon® "Skylake" Scalable Processor is the one of the last generations of Intel®processor implemented the eponym "Skylake" microarchitecture and designed with the same 14nm technology process as its predecessor Intel®Broadwell. However, it has enhanced computing capabilities thanks to a higher core count (Max 28 vs 22), AVX-512 ISA support, a higher memory bandwidth due to an increase by 50% of the memory channels (6 memory channels versus 4) supporting DDR4 up to 2667MT/s and a more balanced memory cache hierarchy compare to prior generations. It also has more IO capabilities thanks to its 48 PCI-e lanes per socket, support PCI-e gen3 and also up to 3 high speed, low-latency, Intel®Ultra Path Interconnect (UPI) links. Intel®UPI technology allows to interconnect processors in a coherent way at high speed (up to 10.4GT/s) to build multi-socket, NUMA systems, up to 8 sockets in glueless mode and more than 8 sockets using proprietary node controller (xNC). Supporting up to 2 (For Advanced RAS SKU Family) AVX-512 FMA (Fused Multiply-Add) execution units, the processor delivers up to 32 double precision floating points operations per cycle, leading to double the number of FLOP/s compared to Broadwell. Skylake support Intel®Hyper-

[2]https://www.spec.org/cgi-bin/osgresults

Threading Technology (Intel®HT Technology) which, when enabled, allows each core to support two threads.

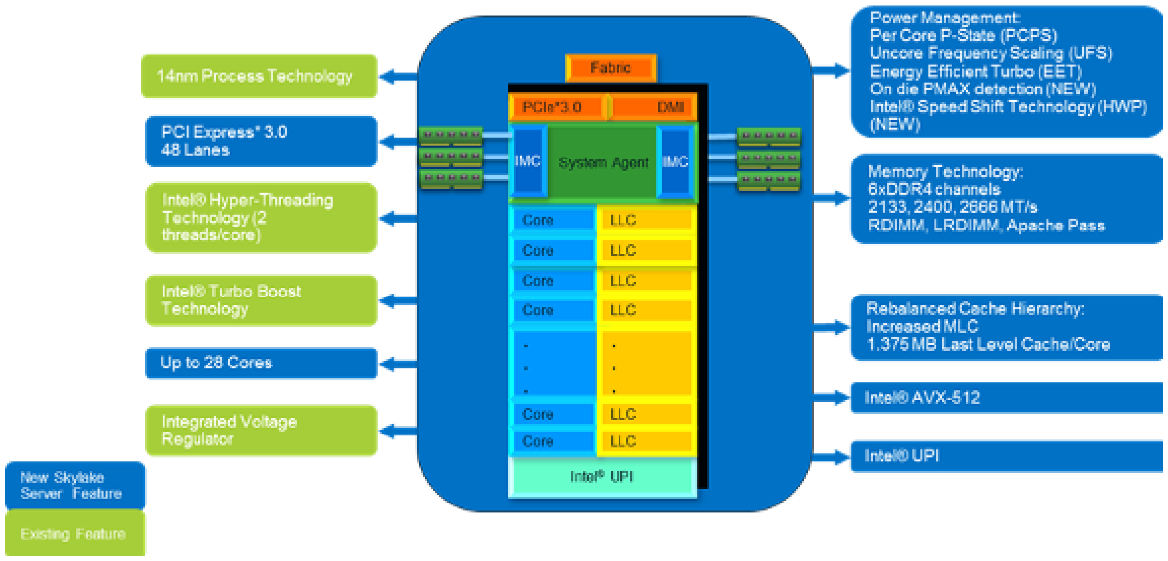

14nm Process Technology
PCI Express* 3.0 48 Lanes
Intel® Hyper-Threading Technology (2 threads/core)
Intel® Turbo Boost Technology
Up to 28 Cores
Integrated Voltage Regulator
New Skylake Server Feature
Existing Feature
Fabric
PCIe*3.0
DMI
IMC
System Agent
IMC
Core
LLC
Intel® UPI
Power Management: Per Core P-State (PCPS) Uncore Frequency Scaling (UFS) Energy Efficient Turbo (EET) On die PMAX detection (NEW) Intel® Speed Shift Technology (HWP) (NEW)
Memory Technology: 6xDDR4 channels 2133, 2400, 2666 MT/s RDIMM, LRDIMM, Apache Pass
Rebalanced Cache Hierarchy: Increased MLC 1.375 MB Last Level Cache/Core
Intel® AVX-512
Intel® UPI

Figure 4: Skylake SP Block Diagram.

Intel® Skylake processor has up to 28 cores, all interconnected through a mesh topology (interconnect-on-die). It implements a more balance caching architecture than its predecessor and an improved Instructions per Cycle (IPC): L1 (FLC - First Level Cache) and L2 (MLC - Mid Level Cache) are dedicated cache per core: FLC is comprised of a 32 KB data cache (DCU) and a 32 KB instruction cache (ICU) while MLC is reaching 1MB which is 4 times larger compared to Broadwell processor. The L3 (LLC – Last Level Cache) is shared between all processors and up to 1.375MB per core. The LLC is divided into multiple slices, cut and distributed on each of the cores. All of the slices are interconnected together to form a single logical cache at the socket level.

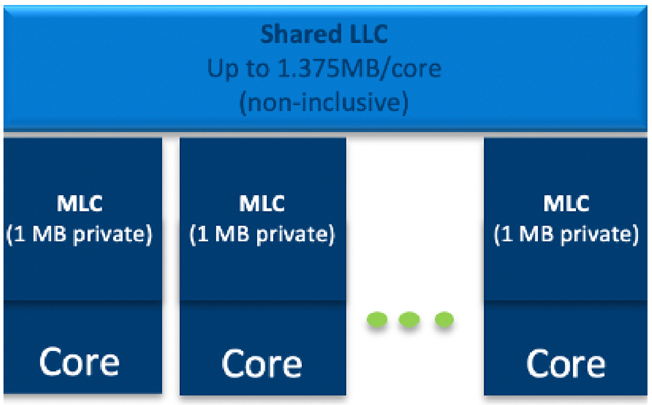


Figure 5: Skylake SP Cache Structure.

The Intel®Xeon®Processor Scalable family will have 4 named shelves, namely, Platinum (81xx), Gold (61xx and 51xx), Silver (41xx) and Bronze (31xx).

The Intel®Skylake SKU (Stock Keeping Unit) deployed initially on the first phase of Joliot-Curie Supercomputer is number 8168. The Intel®Skylake 8168 has 24 cores, running each at 2.7GHZ at the base non-AVX Core frequency, a shared 33MB LLC and a 205W nominal TDP. Each processor is capable to deliver up to It is part 2 TFLOP/s DP Rpeak. It is part of the Platinum shelf which supports advanced RAS features including SMT2, 3 UPI links and 2 AVX512 execution units – which means that each core is capable during a single cycle to execute 2 instructions FMA (Fused Multiply

Add) or 4 double precision operations using a 512 bits vector width[3].

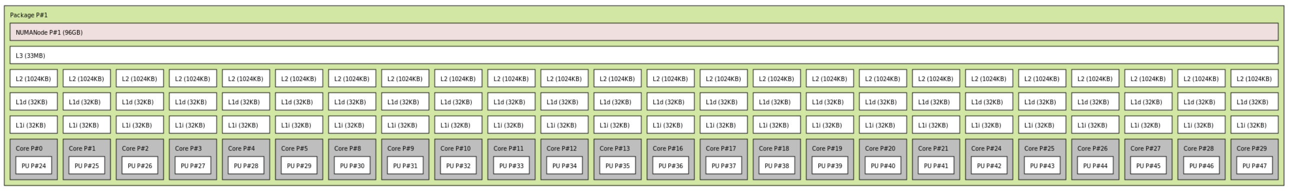
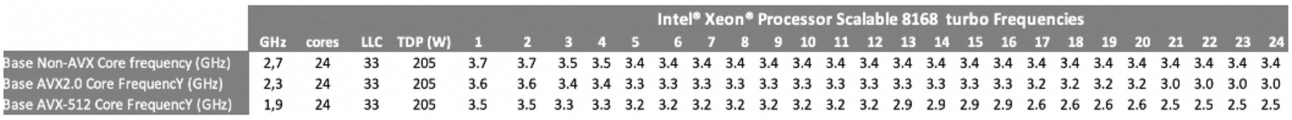


Figure 6: Core topology for the Skylake 8168.

The turbo boost frequency depends of the number of active cores on the SKU and the use of AVX instruction set. As a reminder, here are the frequency values to be expected for the Skylake 8168:

To be finished with node caracteristics.



					Intel® Xeon® Processor Scalable 8168 turbo Frequencies																							
	GHz	cores	LLC	TDP (W)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Base Non-AVX Core frequency (GHz)	2,7	24	33	205	3.7	3.7	3.5	3.5	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
Base AVX2.0 Core FrequencY (GHz)	2,3	24	33	205	3.6	3.6	3.4	3.4	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.2	3.2	3.2	3.2	3.0	3.0	3.0	3.0
Base AVX-512 Core FrequencY (GHz)	1,9	24	33	205	3.5	3.5	3.3	3.3	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	2.9	2.9	2.9	2.9	2.6	2.6	2.6	2.6	2.5	2.5	2.5	2.5

Figure 7: AVX and Turbo Boost Frequencies for the Skylake 8168.

The Intel®Skylake processor like 8168 from the Platinum shelve are dual-socket (2S - 2/3UPI), quad-socket (4S- 2/3UPI), eight-socket 8S(3UPI) servers or even higher (up to 32-socket servers) depending on proprietary NUMA implementation.

For a dual-socket server, like the ones deployed in Joliot-Curie supercomputer, the two 8168 sockets are connected through 2 UPI links. The sustained memory bandwidth as measured with McCalpin Stream benchmark "triad"[4] is around 196GB/s per dual-socket node, i.e around 76.5% memory bandwidth efficiency.

3.4 Marvell ThunderX2

The processor ThunderX2 is the second generation of Marvell 64 bits Armv8-A based processor based on the 16nm process technology. All Marvell ThunderX2 SKUs are part of CN99XX products. It has the capability to scale up to 32 custom Armv8.1 cores running at up to 2.5GHz, supporting also Symmetric Multi-Threading (SMT) but twice number of threads compared to x86 processors like Intel or AMD. The ThunderX2 die is also built on top of monolithic implementation like all the Intel processors up to the Intel Cascade lake generation at least, being the opposite way chosen by AMD to have a MCP (Multi-Chip module implementation). For instance, one ThunderX2 core car support up to 4 threads per core. Each processor supports up to 8 memory channels or 8 memory controllers with up to 2 DPC (DIMM Per Channel), with DDR4 DIMMs running at up to 2667MT/s (1DPC only). The processor has strong I/O capabilities with up to 56 PCI-e gen3 lanes and 14 PCIe controllers along with integrated IO and SATAv3 ports. Each ThunderX2 has a dedicated L1 cache (32 KB instruction and data cache) and a dedicated 256KB L2 cache per core. The L3 cache is 32 MB and is distributed among the 32 cores. In term of computation, the Marvell ThunderX2 supports AVX1 or AVX128 bits and up to 2 FMA execution units which means that each core is capable during a single cycle to execute 2 instructions FMA (Fused Multiply Add) using a 128 bits vector width. This led to one core being capable to run 8 DP floating operations per second.

The Marvell ThunderX2 socket is available either on single- or dual-socket server. For a dual-socket server, like the ones deployed on the INTI cluster, two sockets are connected through Cavium Coherent Processor Interconnect (CCPI2$^{\text{TM}}$) with a 75GB/s theoretical throughput between the 2 ARM sockets. CCPI2 provides full cache coherency within a bi-sockets node.

[3]https://ark.intel.com/content/www/us/en/ark/products/120504/intel-xeon-platinum-8168-processor-33m-cache-2-70-ghz.html

[4]https://www.cs.virginia.edu/stream/

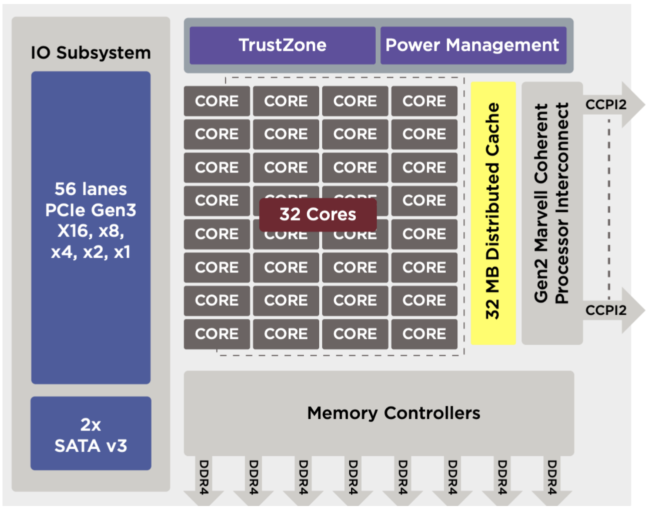


Figure 8: Marvell ThunderX2 Block Diagram.

The Marvell ThunderX2 SKU deployed on INTI ARM prototype is based on ARMv8 architecture with 32 cores, each core running at 2.2GHz, reaching a 0.56 TFlops/s theoretical DP peak performance (Rpeak). It has a theoretical memory bandwidth around 340GB/s, with a sustained stream triad value higher than 241 GB/s, leading to an efficiency around 70%.

4 Results

This section will present the results obtained with the two codes YALES2 and AVBP on the three processors introduced in section 3. The main goal was to enable the characterization of each code on each technology to understand which component (memory bandwidth, NUMA effects, etc.) has an influence on the performance. In particular, the following characteristics have been studied.

- Application's Sensitivity to micro-architecture implementations such as L3 (or LLC) cache access, the memory bandwidth, the inter-processor communication or the raw processor performance.
- Application's Scalability. For YALES2, the main target is of these measurements is the Poisson solver, with a more specific insight on the deflation part.

As a first step towards, the McAlpin STREAM benchmark has been used to help understanding the best placement strategy to implement moreover for applications that are sensitive to memory (memory bound codes).

4.1 STREAM Benchmark

STREAM is the industry standard benchmark for measuring sustained memory bandwidth[5]

Running Stream benchmark on each micro-architecture is part of our methodology to qualify code memory bandwidth dependency, first understanding the best process placement to achieve the best memory bandwidth performance.

4.1.1 AMD EPYC 7H12

The Tab. 1 shows the results of the McCalpin Stream benchmark obtained on the AMD EPYC 7H12 SKU which has a total of 64 cores (1 processor x 64 cores). Single socket configurations with 1, 2, 4, 8, 16, 32 and 64 cores have been tested with 1 thread by core, i.e. without SMT. The results obtained for the 3 sub-tests that involves loads and stores ('Scale', 'Add' and 'Triad') are almost identical and

[5]http://www.cs.virginia.edu/stream/ref.html

lie inside the repeatability variations which were less than 2%. As already mentioned, the EPYC 7H12 SKU is composed of 8 CCD which are grouped in 4 NUMA nodes and each NUMA node shares 2 memory channels. It can thus be anticipated that the thread placement will have a major influence on the measured bandwidth. Two specific placement strategies have been tested in the present work: either *compact* which places the threads on contiguous CPU cores and *scattered* which put the threads as far as possible from one another. In the first case, the first 4 threads are always on the same CCX, the first 8 threads are on the same CCD and the first 16 threads are on the 2 CCD that belong to the same NUMA node. The cases with 32 and 64 threads are then composed of 2 and 4 full NUMA nodes respectively. In the second case, there is only 1 core by CCD up to 8 cores. Then starting from 16 cores all the CCX contains the same amount of active cores (1 by CCX for 16 cores, 2 by CCX for 32 cores and finally 4 by CCX for 64 cores). When the socket is full, both placement strategies are identical.

As expected, the compact and scattered placement strategies provide different results. With the compact strategies, the BW does not increase significantly as long as the active cores remain bind to the first NUMA node (Up to 8): only 2 memory controllers can provide data to these cores while thee bandwidth increased significantly in the scattered mode as all cores (up to 8) benefit each from 2 memory controllers.

The measured bandwidth reaches 169GB/s when 16 cores of a single socket are used in scattered mode, leading to a 82% efficiency of the memory controllers on the 7H12 in comparison to the maximal theoretical bandwidth which is 204.8GB/s. It must also be seen that the bandwidth with 8 and 32 scattered cores is nearly identical. There is a small but noticeable drop in performance when the 64 cores of the socket are used (approximately 5%). This engages to study the behavior of applications which are memory bound when the socket is partially depopulated.

It should also be noted that these results are only achievable when the non-temporal stores are enabled. If not, the maximum achievable bandwidth is only 120 GB/s. At present time, only two compilers provide this option: the Intel (2019.0.5) and the AOCC (2.1.0) suites. Please note that the default behavior of both compilers is different: non-temporal stores are enabled with Intel compiler but disabled with AOCC.

The binaries have been obtained with the following compilation command lines:

```
> icc -DNTIMES=200 -DSTREAM_ARRAY_SIZE=80000000 -O3 -march=core-avx2 -qopenmp stream.
    c -o stream.amd.icc_ntsore_enabled

> icc -DNTIMES=200 -DSTREAM_ARRAY_SIZE=80000000 -O3 -march=core-avx2 -qopenmp -qopt-
    streaming-stores=never stream.c -o stream.amd.icc_ntsore_disabled

> cclang-9 -DNTIMES=50 -DSTREAM_ARRAY_SIZE=80000000 -O3 -march=znver2 -fopenmp -fnt-
    store=aggressive stream.c -o stream.amd.aocc_ntsore_enabled

> cclang-9 -DNTIMES=50 -DSTREAM_ARRAY_SIZE=80000000 -O3 -march=znver2 -fopenmp stream
    .c -o stream.amd.aocc_ntsore_disabled
```

Ncores **compact**	total BW [GB/s]	BW/core [GB/s]	Ncores **scattered**	total BW [GB/s]	BW/core [GB/s]
1	32.0	32.0	1	32.0	32.0
2	39.3	19.7	2	62.7	31.4
4	37.7	9.43	4	123.5	30.9
8	36.3	4.53	8	168.4	21.1
16	40.6	2.54	16	169.0	10.5
32	81.0	2.53	32	167.3	5.22
64	161.3	2.52	64	161.3	2.52

Table 1: Stream Triad McAlpin Benchmark's results - AMD EPYC 7H12 socket

4.1.2 Marvell CN9980-2200

The Tab. 2 shows the results of the McCalpin Stream benchmark obtained on an Marvell CN9980-2200 bi-scokets node which has a total of 64 cores (2 processors x32 cores). A single socket configuration have been tested with 1 thread by core (no SMT) with 1, 2, 4, 8, 16 and 32 cores. Our measurements have confirmed that the thread placement has no influence on the bandwidth on this architecture.

Ncores	total BW [GB/s]	BW/core [GB/s]
1	18.5	18.50
2	34.4	17.20
4	64.2	16.05
8	107.2	13.40
16	114.8	7.18
32	121.4	3.79

Table 2: Stream Triad McAlpin Benchmark's results - Marvell TX2 CN9980-2200 bi-sockets node.

4.1.3 General comments and conclusions

4.2 CFD Benchmarks description

4.2.1 YALES2

The selected benchmark for the YALES2 solver is the well-known PRECCINSTA burner [7, 14, 9, 12, 1]. It is a semi-industrial burner which is relevant for the aeronautical combustion applications. In the present study, it is operated in isothermal conditions: only the constant-density flow solver will be used. This benchmark is available on 3 different meshes (noted here M1, M2 and M3) which contains respectively 14M, 110M and 878M cells. Each mesh is actually obtained from the previous one by performing an homogeneous mesh refinement. In all cases the cell-group size has been set to 2'000 which is very close of the optimum for all the platforms. All the data presented below are obtained for a simulation of 200 time-steps: depending on the mesh size, the targeted machine and on the number of cores, the simulation time was between 1 minute and 30 minutes which is sufficient to obtain significant statistics. All the presented data have been obtained by the embedded timers of YALES2 which provide a flexible instrumentation of specific parts of the code. These embedded timers are based on the RDTSC instruction on x86_64 and MPI_WTIME on ARM architectures.

4.2.2 AVBP

Initial performance tests for characterization and porting were performance using a 3M cells PRECCINSTA case. The selected benchmarks for strong scaling for AVBP are the BKD burner from DLR [6] consisting of 42 injectors used in a collaboration between Thomas Schmitt (EM2C-CNRS) and Cerfacs on a current grand challenge[15] in the IRENE Rome system using a 1.4B tetrahedras mesh and a 1B tetrahedras turbulent channel. For weak scaling a simple turbulent channel case was use with XXX meshes (1.2M, 10.2M, 82.2M and 314M respectively). Each mesh is obtained from the previous one using a uniform refinement constraints and the MMG library [3]. Measurements where performed at least 5 times to evaluate variability and each simulation last between 30s to 5min. The results reported here are the worst case scenario observed. The data was obtained by embedded timers using MPI_WTIME.

4.3 Code characterization

4.3.1 Best practices

4.3.2 YALES2

Porting the YALES2 code to AMD was straightforward. The compilation options are listed below:

Intel:

- -align array64byte -align rec32byte -unroll-aggressive
- export I_MPI_FABRICS=shm:ofa

AMD:

- -align array64byte -align rec32byte -unroll-aggressive
- export MKL_DEBUG_CPU_TYPE=5
- export I_MPI_FABRICS=shm:ofa

TX2:

- stream: armclang -DSTREAM_ARRAY_SIZE=100000000 -O3 -fopenmp -march=armv8.1-a -mcpu=native -fsimdmath -ffp-contract=fast stream.c -o stream

The default parallel environment is the following:

- openmpi/4.0.2 (default options).
- ucx/1.7.0
- hcoll/4.4.2938

4.3.3 AVBP

Porting the AVBP code to AMD was straightforward. To all for universal binaries since the filesystem of IRENE Rome is shared with the IRENE Skylake and IRENE KNL platforms, the *-O3 -mavx2 -axCORE-AVX2 -fma* compiler flags were use with the intel compiler 2019.0.5.281. The default environment for our tests unless stated otherwise is:

- openmpi/4.0.2 (default options).
- ucx/1.7.0
- hcoll/4.4.2938

4.3.4 Methodology

Contrary to the Intel and Thunderx architectures, process placement on the CCX and CCD of an AMD processor can yield dramatic consequences for performance of a code. Indeed as described in the previous sections, L3 cache and memory bandwidth are not uniform for all process. In order to characterize the dependency of the code to memory bandwidth it is possible to place processes so as to double the available L3 cache or memory bandwidth thus measuring their importance of the code. Using the script bellow, we can control the stride and group size of process placement within a CCX and CCD. For example ...

```
export OMPI_MCA_hwloc_base_report_bindings=1

# binding options
GROUPSIZE=2
STRIDE=4
ENDCPU=127
LISTCPU=$( for i in $(seq 0 $STRIDE $ENDCPU);do printf $( seq -s"," $i 1 $(( $i+${
    GROUPSIZE}-1)) );printf ",";done |sed 's/.$//' )
MY_SLURM_OPTS="--ntasks-per-node=$NTASKS_PER_NODE --cpu-bind=verbose,map_cpu:$LISTCPU
     --distribution=block"

ccc_mprun -l -E "$MY_SLURM_OPTS" ./exec
```

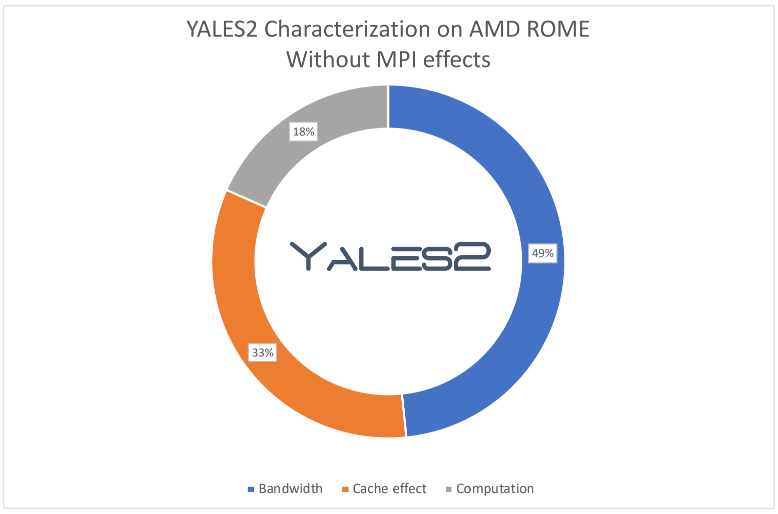


Figure 9: Characterization of YALES2 on AMD Rome architecture.

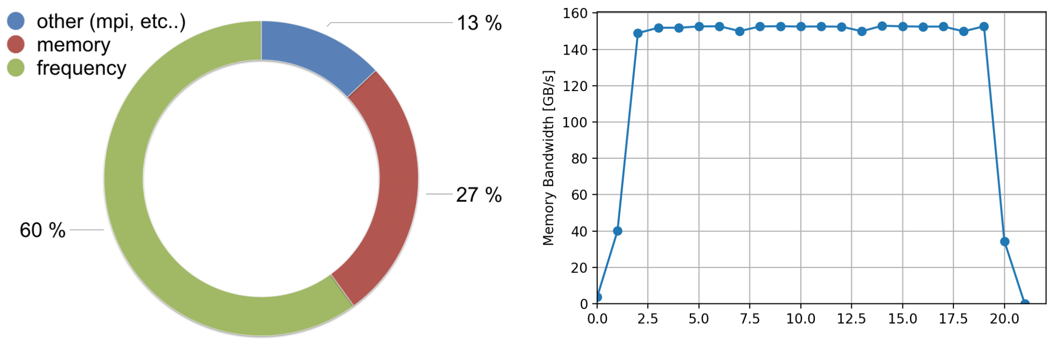


Figure 10: left: Characterization of AVBP sur architecture AMD. Right: Measured bandwidth during the simulation

4.3.5 YALES2

The Fig. 9 present the results obtained on the YALES2 code, following the procedure explained above. Only the cases labeled 3, 4 and 5 have been used to obtain the presented results.

4.3.6 AVBP

Code characterization using the placement strategy described previously is shown in figure10. It suggests that AVBP is compute bound (60%) which concurs with developers experience who know well the influence of CPU clock speed on simulation times. This is also confirmed using the turbo mode (which is enabled by default on the Joliot-Curie AMD Rome partition for all these tests). Disabling it (via the module feature/system/turbo_off provided by the system) reduced AVBP code's efficiency by 10% in average with a turbo mode frequency sets to off. Bandwidth measurements during runtime were performed using AMDuProf tool. [6]. A max usage of 150GB/s confirms that AVBP does not saturate the available bandwidth of the node (409,6GB/s). The influence of memory bandwidth on AVBP is around 27%. This confirm that AVBP is not Bandwidth driven. However the moderate impact shows that the code will be sensitive to the core count and process placement. The architecture will impact performance if the core count continues to increase and bandwidth does not.

[6]https://developer.amd.com/amd-uprof/

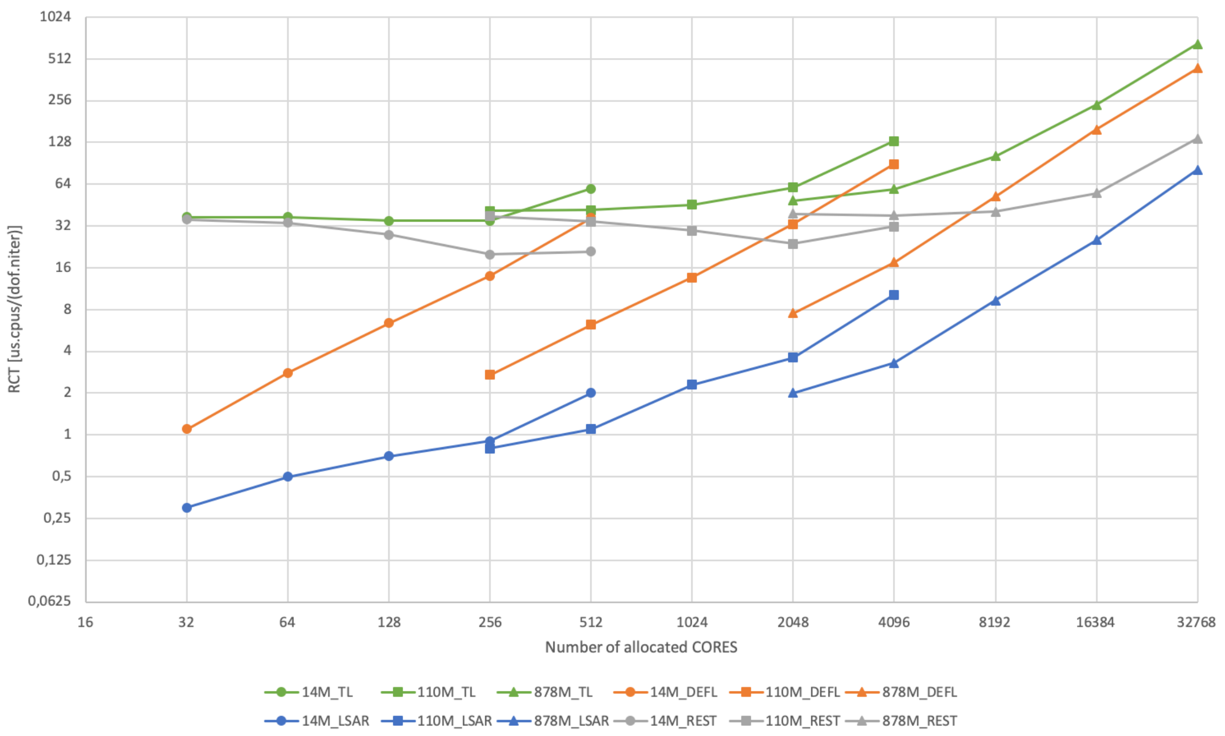


Figure 11: Scaling performance for YALES2 on Joliot Curie AMD for the PRECCINSTA burner at various resolutions. TL: total, DEFL: linear solving of the Poisson equation with the DPCG, LSAR: Allreduce communications in the linear solver (included in DEFL), REST: rest of the temporal loop.

4.4 Parallel performances

4.4.1 Methodology

4.4.2 YALES2

The YALES2 strong and weak scaling measurements are presented in Fig. 11. This figure gives the Reduced Computational Time, i.e. the time spent in the temporal loop per iteration and per node. This number is normally constant for an ideal speed-up. The different curves of the figure represent the different parts of the temporal loop. It can be observed that the weak scaling, i.e. with a constant load per core is good. The strong scaling however is not ideal.

4.4.3 AVBP

The AVBP strong scaling measurements were performed using openmpi 4.0.2 and flat MPI (1 MPI per core) on full nodes (128 cores). Domain decomposition mapping where pre-computed to avoid collective operations crashes observed with Parmetis on more than 4000 mpi tasks. Measurement include the full temporal loop of the code but do not account for the initialization phase which takes up less than 1% of the compute time of any simulation and therefore is not relevant. No online co-processing was used during this tests and only a final check-pointing file was written in the finalization phase. I/O is not included in the timing measurements. For weak-scaling, the same setup applies. Additionally, intelmpi 2019.0.5.281 was also tested as well as under-subscribing the nodes with only one core over two used. Figure 12 shows the results. Strong scaling is excellent for both use cases. Weak scaling shows a very good behavior up to 32k cores. The increasing trend in RCT is explained by the duplicated nodes in the parallel interfaces. Overall openmpi 4.0.2 offers a better performance than intelmpi in this system (Disclaimer : intelmpi parameters have not been optimized for this tests). On one node, openmpi is 10% faster, on 250% almost 50% when running on full nodes. Under-subscribing improves RCT by 30% and this constant even if we increase the number of nodes and mesh size. Given the strong scaling performance of the code at full nodes, we only recommend under-subscribing if mpi load in-balance is over 30%.

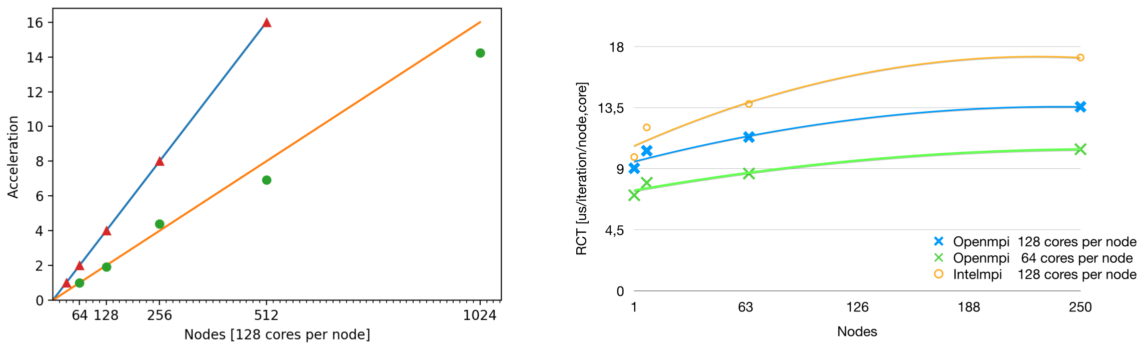


Figure 12: Left: Strong scaling performance for AVBP on Joliot Curie AMD. Turbulent channel LES (circle); Reactive rocket engine LES (triangles) versus respective ideal acceleration (line). Right: Weak scaling for a turbulent channel LES versus MPI library and process placement

.

5 Conclusion

6 Acknowledgments

We would like to thank GENCI and TGCC/ CEA for their support by providing the CFD community 850'000 computing hours on the Tiers 0 Joliot-Curie Supercomputer built on top of 2292 AMDx86 7H12 dual-socket nodes embedded in an extremely high energy efficient Atos/Bull Sequana XH2000 Platform. The research regarding AVBP leading to these results has received funding from the European Union's Horizon2020 research and innovation program under the EPEEC project, grant agreement No 801051 as well support by EXCELLERAT project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823691.".

References

[1] P. Benard, G. Lartigue, V. Moureau, and R. Mercier. Large-Eddy Simulation of the lean-premixed PRECCINSTA burner with wall heat loss. *Proceedings of the Combustion Institute*, 000:1–11, 2019. ISSN 15407489.

[2] M. Bernard, G. Lartigue, G. Balarac, V. Moureau, and G. Puigt. A framework to perform high-order deconvolution for finite-volume method on simplicial meshes. *International Journal for Numerical Methods in Fluids*, in press(n/a), 2020. doi: 10.1002/fld.4839. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4839`.

[3] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358 – 378, 2014. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2014.01.005. URL `http://www.sciencedirect.com/science/article/pii/S0021999114000266`.

[4] F Ducros, F Nicoud, and T Poinsot. Wall-adapting local eddy-viscosity models for simulations in complex geometries. *Numerical Methods for Fluid Dynamics VI*, pages 293–299, 1998.

[5] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics (1989-1993)*, 3(7):1760–1765, 1991.

[6] Stefan Gröning, Justin Hardi, Dmitry Suslov, and Michael Oschwald. Influence of hydrogen temperature on the stability of a rocket engine combustor operated with hydrogen and oxygen. *CEAS Space Journal*, 9(1):59–76, Mar 2017. ISSN 1868-2510. doi: 10.1007/s12567-016-0130-8. URL `https://doi.org/10.1007/s12567-016-0130-8`.

[7] G. Lartigue, U. Meier, and C. Bérat. Experimental and numerical investigation of self-excited combustion oscillations in a scaled gas turbine combustor. *Applied Thermal Engineering*, 24 (11-12):1583 – 1592, 2004.

[8] M. Malandain, N. Maheu, and V. Moureau. Optimization of the deflated conjugate gradient algorithm for the solving of elliptic equations on massively parallel machines. *J. Comp. Physics*, 238:32–47, 2013. URL `http://dx.doi.org/10.1016/j.jcp.2012.11.046`.

[9] W. Meier, P. Weigand, X.R. Duan, and R. Giezendanner-Thoben. Detailed characterization of the dynamics of thermoacoustic pulsations in a lean premixed swirl flame. *Combustion and Flame*, 150(1-2):2 – 26, 2007.

[10] V. Moureau, G. Lartigue, Y. Sommerer, C. Angelberger, O. Colin, and T. Poinsot. Numerical methods for unsteady compressible multi-component reacting flows on fixed and moving grids. *J. Comp. Physics*, 202:710–736, 2005. URL `http://dx.doi.org/10.1016/j.jcp.2004.08.003`.

[11] V. Moureau, P. Domingo, and L. Vervisch. Design of a massively parallel cfd code for complex geometries. *Comptes Rendus Mécanique*, 339(2-3):141–148, 2011. URL `http://dx.doi.org/10.1016/j.crme.2010.12.001`.

[12] V. Moureau, P. Domingo, and L. Vervisch. From large-eddy simulation to direct numerical simulation of a lean premixed swirl flame: Filtered laminar flame-pdf modelling. *Comb. and Flame*, 158:1340–1357, 2011. doi: doi:10.1016/j.combustflame.2010.12.004. URL `http://dx.doi.org/10.1016/j.combustflame.2010.12.004`.

[13] Franck Nicoud, Hubert Baya Toda, Olivier Cabrit, Sanjeeb Bose, and Jungil Lee. Using singular values to build a subgrid-scale model for large eddy simulations. *Physics of Fluids*, 23(8):085106, 2011.

[14] S. Roux, G. Lartigue, T. Poinsot, U. Meier, and C. Berat. Studies of mean and unsteady flow in a swirled combustor using experiments, acoustic analysis, and large eddy simulations. *Comb. Flame*, 141(1-2):40–54, 2005.

[15] T. Schmitt, G. Staffelbach, S. Ducruix, S. Gröning, J. Hardi, and M. Oschwald. Large-eddy simulations of a lab-scale liquid rocket engine: influence of fuel injection temperature on thermo-acoustic stability. In *7th European Conference for Aeronautics and Space Sciences (eucass 2017)*, Italy, 2017. Politecnico Milano.