

**ExCALIBUR   
  
Benchmarking Requirements for NEPTUNE and associated tools  
  
M3.5.1**

**Abstract**

This report describes the work done for NEPTUNE project at Milestone 3.5.1 for   
the deliverable D3.1 “Verification and benchmarking methodology”. In this   
document we present a short characterisation of testing for   
Computational Science and Engineering applications and   
preliminary benchmark results for the   
AMD Rome processor.

**UKAEA REFERENCE AND APPROVAL SHEET**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Client Reference: | |  | |
| UKAEA Reference: | | CD/EXCALIBUR-FMS/0016 | |
| Issue: | | 1.00 | |
| Date: | | 10 March 2020 | |
| Project Name: ExCALIBUR Fusion Modelling System.  Version 1.00 | | | | | |
|  | Name and Department | | Signature | | Date |
| Prepared By: | Lucian Anton | | N/A | | 12/03/2020 |
| Reviewed By: | Rob Akers | | N/A | | 12/03/2020 |
| Modified By: | Rob Akers  MSSC | |  | | 12/03/2020 |
| Approved By: | Rob Akers  MSSC | |  | | 12/03/2020 |

# Introduction

In the information technology industry it is acknowledged that there is a general “verification gap problem”, that is, hardware and software verification efficiency is lagging behind software productivity and microprocessor component number/complexity growth [1]. However, this activity cannot be neglected because it is crucial to deliver the quality of the candidate systems. In order to mitigate this problem, industry has placed verification and validation methodologies at the core of its processes.

For Computational Science and Engineering (CSE) applications, accumulated experience has shown that methodical testing is needed at “all stages” of the software development process, as the complexity of applications and of the associated processing hardware continue to increase year upon year.

Regular and extensive testing is very important for the following reasons [2]:

* It promotes high quality software,
* Doing so increases the quality and speed of development,
* Extensive testing maintains portability and
* facilitates refactoring and the addition of new features.

In CSE, tests can be categorised according to their granularity: unit tests, integration tests and system level tests or according to their type: verification tests, acceptance (validation) tests, no-change or characterization tests and performance tests.

Sets of tests are used to build regression test suites which check if a code has not lost capabilities or behaviour following a source or infrastructure (such as: hardware, compilers, libraries) change. A “non-regression test suite” is a set of new tests for new functionalities.

High quality testing requires one to partition the testing activity into several roles:

* **verification and validation tester**: an expert in modelling and numerics,
* **performance tester**: an expert in HPC,
* **maintenance tester**: responsible for the test suites themselves
* **manager tester**: an individual who has overall responsibility for testing, including coverage, policies, quality of verification and validation, testing for production validation, etc .

Role allocations are flexible and can depend upon the size of the development team. An individual could occupy several roles, or one could have a team covering one role.

Testing activities require extra time to identify and analyse a test and the associated development time. In order keep the costs of testing down, automation needs to be used wherever possible. Nowadays this is relatively is easy to do in a standard way using continuous integration tools. The UKAEA GitLab repository offers to new projects a skeleton repository for C++, Fortran or python based projects which have unit testing fully integrated.

For NEPTUNE proxy-apps and the later software components, we plan to use a test driven development methodology to ensure efficiency and quality of the process.

Performance testing (or benchmarking) occupies a special role in NEPTUNE as it has to deliver software for exascale class systems. In order to avoid building a system with poor performance, performance testing will follow the granularity used for verification: unit, integration, system. The year 2 activities for NEPTUNE will focus upon ensuring that node level and system level performance is explored with the proxy apps and that realistic targets are set for the performance critical components in the second stage of the project. Besides speed metrics (such as time to solution, degrees of freedom / s) the convergence rate of solvers, load balancing of the generated mesh will be assessed together with any other features of the algorithm which related to parallel performance. The performance of the proxy-apps or test codes will be also assessed on the new computational hardware which will become available through the Isambard 2 project and ExCALIBUR hardware allocation.

# Preliminary benchmark results for AMD Rome processor

In this section we present a brief comparison of the performance characteristics between the new AMD Rome processor and the Intel Skylake processor, as an example of the type of performance testing we plan to be ubiquitous throughout the project. Results were obtained with the simple test code JTC (Jacobi Test Code) [3] described below and the gyrokinetic code GS2 [4].

Benchmark runs were carried out on AMD Rome single node systems which were made available at the University of Durham for a short period of time at the end of 2019 and on the Skylake nodes of the Marconi system available at CINECA HPC centre.

## **Brief description of the tested processors**

We acquired access to two types of AMD Rome processor, with 32 and 64 cores/socket, each test node had 2 sockets. Their main characteristics are described in Table 1 together with the characteristics of the Skylake processor. One distinguishing feature of the AMD Rome architecture are the chiplets, groups of 8 cores that share two banks of 16 MB L3 cache memory. More details around the AMD Zen2 architecture can be found in Refs [5] [6] [7].

|  |  |  |  |
| --- | --- | --- | --- |
|  | Skylake | AMD Rome | |
| model | Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz | AMD EPYC 7452 32-Core Processor | AMD EPYC 7702 64-Core |
| cores/socket | 24 | 32 | 64 |
| L1 | 32K | 32K | 32K |
| L2 | 1024K | 512K | 512K |
| L3 | 33792K | 8 x 16384K | 16x16384K |
| Clock speed | 2100 | 2345.584 | 1996.149 |

*Table 1: Main features of the tested processors.*

## **Performance profiles with Jacobi Test Code (JTC)**

JTC implements a 6 points stencil iteration over a 3D cuboid domain. The algorithm is implemented in mixed mode (MPI+OpenMP), OpenCL and CUDA, so it can be used to measure the performance of one algorithm across several architectures.

We have carried out runs on the three available processors with executables built with Intel and GNU compilers for the AMD processors and only with an Intel built executables for Skylake. All runs were carried out in pure MPI mode.

A close up of a map

Description automatically generated

Figure 1: Average number of grid point updates per second as a function of the linear grid size for runs performed with Intel compiler executables on the three processors described in the text. The first section of the plot, which varies strongly upon the grid size, shoes the compute bound feature of the kernel, the plateau sector is a measure of the bandwidth to main-board memory.

In Figure 1 one can see that the larger total cache memory and the higher number of cores of the AMD processors have a significant impact upon performance for grids which fit into cache memory. The ratio of the peak performance between Rome and Skylake is approximately 2.4 and 3.4 for Rome with 64 and 128 cores, respectively. This compares very advantageously for Rome processors as the ratio for the number of cores are 1.3 and 2.7 respectively. There is a caveat in this comparison - peak performance occurs at a different number of cores for different processors. But even if we use a grid size at which Skylake performance is optimal, the Rome processors have good ratios: ~2 and ~2.6 respectively. We note also that the Rome node with 128 cores delivers diminishing returns, that is, its peak performance is not close to twice the performance of the 64 core node.

In the plateau region, the AMD processor has a larger memory bandwidth close to memory channel ratio 4/3 between Rome and Skylake processors. We have repeated this test also with the executable generated with the GNU compiler with similar results.

In the second benchmark we have run a linear benchmark of the gyrokinetic code GS2 on a representative input. We show in Table 2 the wall-clock time for the first two sectors reported by the code internal timer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AMD Rome | | | | Skylake |
| Compiler | Intel 2019u4 | | GNU 7.3.0 | | Intel 2018 |
| Cores | 32x2 | 64x2 | 32x2 | 64x2 | 24x2 |
| Advance step | 2.63 | 2.64 | 4.26 | 3.34 | 3.77 |
| Fields solve | 0.215 | 0.333 | 1.034 | 0.614 | 0.497 |

*Table 2: Wall-clock time spent in two sections of GS2 (“Advance step” and “Field solve”) for the Rome and Skylake processors. In the case of the Rome processor, data were obtained with an executable built with Intel and GNU compilers, only an Intel compiled executable was available for Skylake.*

One can see that for Rome, the Intel compiler produces a significantly faster executable, however there is no performance increase when using the 128 core node. This result would need further investigation. Comparing the Rome 64 vs Skylake 48 speed ratios vs. number of core ratio one finds that Rome delivers a slight advantage in the “Advance step” section and dominates in the “Field section”.

Tests such as this will not only be useful for determine the optimal candidate hardware architectures for the NEPTUNE referent models, but will also be invaluable for uncovering the data-flow / performance bottlenecks of the algorithms as they are developed. Verification and performance benchmarking activities will thus form a major element of the project, and will be specified as part of all relevant 3rd party defrayed activities.

# Bibliography

|  |  |
| --- | --- |
| [1] | I. Philips, “Stronger than its weakest li link,” in *High Integrity So.ware Conference (HIS'15)*, Bristol, 2015. |
| [2] | R. A. Bartlett, A. Dubey, X. S. Li, J. D. Moulton, J. M. Willenbrig and U. M. Yang, “Testing of Scientific Software: Impacts on Research Credibility, Development Productivity, Maturation, and Sustainability,” in *Software Engineering for Science*, CRC Press, 2017, pp. 89-118. |
| [3] | L. Anton, M. Mawson, A. D. Taylor and V. Szeremi, “Performance profiles with Jacobi test Code,” 2015. |
| [4] | “GS2,” [Online]. Available: https://bitbucket.org/gyrokinetics/gs2/src/master/. |
| [5] | J. D. Gelas, “CPUs AMD Rome Second Generation EPYC Review: 2x 64-core Benchmarked,” August 2019. [Online]. Available: https://www.anandtech.com/show/14694/amd-rome-epyc-2nd-gen/2. |
| [6] | D. I. Cutress, “AMD Zen 2 Microarchitecture Analysis: Ryzen 3000 and EPYC Rome,” June 2019. [Online]. Available: https://www.anandtech.com/show/14525/amd-zen-2-microarchitecture-analysis-ryzen-3000-and-epyc-rome/. |
| [7] | J. D. Gelas, “AMD Rome Second Generation EPYC Review: 2x 64-core Benchmarked,” August 2019. [Online]. Available: https://www.anandtech.com/show/14694/amd-rome-epyc-2nd-gen/4. |