Introduction

The tools described in the previous part for comprehension of High Performance Computing from theory, hardware and software give us the basis elements to go toward optimizations and benchmarking. We showed through examples that hybrid architectures seem to be the way to reach exascale in few years. In the same time many optimizations need to be done. On one hand, the exascale supercomputer will need to fit the energy envelope imposed, to be able to sustain the computational power and the price of the hardware itself for economical reasons. This is made by vendors and is a specific area of optimization. On the other hand, we focus on the performances we obtained using a specific architecture/accelerator to fit the targeted application with the best resources usage. The need of regular memory accesses, synchronization and the host-device memory separation put some constraint on their usage. As many-core architectures presented nowadays come with several downsides we need to confront them to classical processors on a specific set of problems, a metric, a dedicated benchmark suite.

In benchmarks like the TOP500 the target is to solve a problem with regular computation and communication behavior. We think that this behavior does not fit realistic and production applications. In many domains like meteorology, oceanography, astrophysics, big data, ... the underlying issue are the irregular input and behavior. The irregularity in an application can have several definitions. This is defined by [JTB] as a problem which: can not be characterize a priori, is input data dependent and evolves with the computation itself. In [SL06] the author specifies that the work involves subcomputations which cannot be determined before and implies work distribution during runtime. The irregularity can then spread on all the layers of the resolution: the communications, the computation and the memory searches.

In order to target more representative applications of nowadays realistic problems, we identify several bottlenecks and limitations in HPC. Those limitations are called *walls* against which nowadays architectures are confronted to reach exascale.

Memory Wall: This problem was targeted for the first time in [WM95]. The authors explain that:

> We all know that the rate of improvement in microprocessor speed exceeds the rate of improvement in DRAM memory speed, each is improving exponentially, but the exponent for microprocessors is substantially larger than that for DRAMs.

In the case of accelerators another layer of memory is added. The memory of the host processors and device accelerators cannot be accessed directly and copies from one to the other are requested. The problems of coalescent accesses are also addressed in this study. Some companies try to find ways using shared memory between host and device but this technology is always under development and test for HPC purpose.

The two problems we propose for our metric implement heavy memory utilization. The first one dynamically uses the memory in an irregular way. The tree traversal of the first method generates temporary data and binary tests. The algebraic method uses a dedicated big integers library and carries propagation is applied just when necessary. In the second benchmark the memory is saturated and prepared at the startup. It is then accessed in an irregular way during all the computation.

Communication wall: We showed that the supercomputers' architecture is based on a set of racks, composed of nodes composed of computation units. The network topology can never be perfect for all the kinds of problems and even the fastest technologies are limited to the software handling. Limiting the big synchronization steps, like in the BSP model, allows the system to be asynchronous and hide computation by communications. Unfortunately this is not applicable to all the applications and a huge care have to be taken to approach perfect scaling.

We decided to target this problem in two main ways. In the first benchmark the model corresponds to FIIT: Finite number of Independent and Irregular Tasks, introduced in [FKF03]. The tasks can be solved independently and then merged at the end. We show that the accelerators can also take advantage of specific distribution methods like Best-Effort. In the second problem communications are central because the data are too big to be represented on a single machine and cannot be computed independently. The irregularity in communication is very high. In this benchmark the amount of data shared is never known before reaching the end of a superstep.

Power wall: The energy consumption of nowadays and future supercomputers is the main wall in HPC. Indeed, an exascale supercomputer could be built using several petascale supercomputers but, with todays architectures, will require a full nuclear plant to operate. In this objective low energy consumption and innovative architectures need to be found. The energy considered is required to power the machine itself but also handle the heat generated.

This wall is the underlying goal of this study. The hybrid architectures seem to deliver better performances for less watts. This thesis shows through different benchmarks what the real benefit of accelerators is on realistic problems and that the performances can be even better for less watts of power consumption.

Computational wall: The computational wall is a combination of the wall presented before. By increasing the memory wall, the energy consumption and the communications we can increase the overall computation power of the supercomputer. The limitation in computational power also comes from the fact that the Moore's law seems to be over. Vendors have more difficulties to shrink transistors due to physical side effects. The frequency itself seems to reach its highest values due to the energy required to operate, the heat dissipated and synchronization issues.

This is targeted in the first benchmark we propose. The algebraic method is heavy in computation with irregular memory accesses. The second one does focus on irregular communication and memory usage with test based operations.

This is the reason why we propose in this part a new metric for the main HPC walls to confront many-core architectures to multi-core architectures. The two applications we targeted cover all the walls we specify.

The first academic problem is a perfect candidate for our benchmark since it characterizes the behavior of accelerators and more specifically GPUs focusing on irregular computations and memory accesses. It is the problem of Langford and it shows how we can take advantage of accelerators for this kind of problem. Several paper were published on this problem [**?**, **?**, **?**, **?**]

The second problem focuses on irregular memory accesses and communications. It is based on a benchmark we presented in the previous part, the Graph500 benchmark. This problem was presented to the community with [**?**, **?**, **?**].

The combination of those two problems creates a metric covering all the walls and limitations we are facing in HPC. This allows us to emphasis on the behavior of hybrid architecture on several type of problems.

Bibliography

[FKF03] Olivier Flauzac, Michaël Krajecki, and Jean Fugère. Confiit: a middleware for peer to peer computing. In *International Conference on Computational Science and Its Applications*, pages 69–78. Springer, 2003.

[JTB] Christopher J Riley High-Performance Java and Gabrielle Keller Transformation-Based. Irregular parallel algorithms.

[SL06] Michael Süß and Claudia Leopold. Implementing irregular parallel algorithms with openmp. In *European Conference on Parallel Processing*, pages 635–644. Springer, 2006.

[WM95] Wm A Wulf and Sally A McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.