Setting up an Operational Academic HPC Pole

Claude Tadonki
Mines ParisTech - PSL Research University
Centre de Recherche en Informatique (CRI)
35, rue Saint-Honoré, 77305, Fontainebleau Cedex (France)
Email: claude.tadonki@mines-paristech.fr

***Abstract*—Hardware support for high-performance computing (HPC) has so far been subject to significant advances. The pervasiveness of related systems, mainly made up with parallel computing units, makes it crucial to spread and vivify effective HPC curricula. Besides didactic considerations, it appears very important to implement HPC hardware infrastructures that will serves for practices, and also for scientific and industrial requests. The latter ensures a valuable connection with surrounding cutting-edge research activities in other topics (*life sciences, physics, data mining, applied mathematics, finance, quantitative economy, engineering sciences*, to name a few), and also with industrial entities and services providers from their requests related to HPC means and expertise. This aspect is very important as it makes an HPC pole becoming a social actor, while bringing real-life scenarios into the academic context. The current paper describes the major steps and objectives for a consistent HPC curriculum, with specific analyses of particular contexts; suggests how to technically set up operational HPC infrastructures; and discusses the connection with end-users, all these in both effective and prospective standpoints.**

I. Introduction

With the advent and pervasiveness of multicore processors, *parallel processing* is continuously getting out of mystery. However, even if it is true that most of end-users of common applications are now somehow aware of their parallel execution, and that ordinary programmers are ready or more willing to admit the necessity of parallel processing considerations, delving into *parallel computing topics* [9], [17], [18], [25], [6], [8], [16] is still not an instinctive nor intuitive step. People need to be convinced about that choice, and the balance between the underlying effort and the corresponding reward should appear favorable. Thus the need of incentive HPC initiatives.

Significant efforts have been made so far to make *parallel computing* implementation as easy as possible, if not seamless from the programming viewpoint [14], [15], [12], [23], [24]. However, the topic is still non trivial [2], [7]. To date, there is no effective compiler who can automatically derive a parallel version of a given ordinary sequential code without explicit user annotations or monitoring. Indeed, *parallel computing* adds another abstraction to *basic algorithm design* that has nothing to do with the underlying problem being solved. Thus, the topic might be perceived as a plain complication, otherwise technically speculative. Thus, we need to grab people attention and make them accept to take the way of *parallelism*. This is an education goal, prior to targeting an increasing number of (potential) candidates for parallel computing curriculum.

There is a increasing need of programmers with parallel programming skills. Thus, it is now common to see job ads seeking people qualified or specialists in *parallel computing*. This has significantly motivated *parallel computing curricula*, which, from our point of view, should not stand totally disconnected from other areas. Indeed, it is always better to understand what we have to parallelize. This will result in a more skillful parallel scheme and yield a consistent self-contained report. Thus, it is preferable to have good basic skills before moving to parallelism. This through has to be kept in mind while designing an HPC curriculum.

From a technical point of view, especially with *shared memory parallelism* [3], there is a heavy hardware concurrency, which should be managed or taken into account carefully at design or evaluation time [22]. Thus, it is important to make sure that the *basis of computer architecture* [19], [21] are well understood, maybe making it a prerequisite. Another reason for this is the impact that a parallel scheme might have on *numerical aspects*. Also for *proof of correctness*, hardware mechanisms of parallel systems should be included into the equation. Unfortunately, it is very common to encounter *computer architecture unaware* parallelizations. This might be due to a lack of related skills or just a plain carelessness. The most noticeable example is the important proportion of programmers who are completely unaware of *vector computing*, also referred as *SIMD computing*, whose related hardware elements and mechanisms are implemented in almost all modern processors. Thus, providing a computer architecture course at the earliest stage is certainly a good idea.

We see that there are numerous facts and thoughts that emerge when it comes to *parallel computing* from a didactic viewpoint. Thus, we need to efficiently organize and manage all related teaching/training/expertise activities [28]. Beside invariant common needs, there are specific contextual issues that need to be addressed conscientiously through well targeted applications. Within an academic context, the best way to handle all these initiatives is to set up an HPC pole that will implement the main didactic actions and promote the use of *parallel computing* as previously stated. The purpose of this paper is to provide a consistent panoramic view of an academic HPC pole including the key topics with their interaction, and to describe its main associated activities. The rest of the paper is organized as follows. The next section describes key didactic factors related to HPC, followed by an overview of how to design and implemented an HPC pole.

Section IV addresses the question of building a local HPC cluster. Section V concludes the paper.

II. Didactic key factors about HPC

A. Justify why considering parallel computing

First of all, we have to sell the need for HPC [1], [26], [30] to our (potential) audience. This preliminary communication is crucial, because people need to be convinced that the related investment is worth considering. The requested effort is not only technical, but also conceptual, as it implies a change of mind about computing devices, programs, algorithms and programming. This should be a clear winning step in order to capture a sincere and keen attention of the listeners, their enthusiasm will determine the success of the course and associated exchanges. We should always keep in mind that ordinary people expect the increasing CPU frequency to fit their needs for speed. The way to achieve the aforementioned marketing goal is to raise a number of (intentionally *pro domo*) arguments. We state some of them:

- There are numerous important real-life applications where processing speed is essential, but which cannot be fulfilled at the expected level by a single CPU [10], [5], [29]. Nice and illustrative examples should be described here, some of them following an anecdotal way if possible. Complicated cases should be avoided, even if (and this can be at most mentioned) *scientific computing* is the major "customer" of parallel computing [11], [**?**], [14]. We suggest for instance: *urgent query in a large distributed database*; *real-time high definition image processing*; *realistic high precision simulation*; *new generation video gaming*. With a scientific audience, we use to display figure Fig.1 with *difficult network graphs problems* as examples.

Fig. 1. Computing need standing over computers capability

- The so-called *Big Data* is an active topic for good reasons. Indeed, the volume of data to be processed in number of real-life applications is more and more huge, and the response is expected to remain almost instantaneous whatever the scenario. The case of social networks is a good example here. Data analytics applications can be also mentioned. The major actors of the two previous topics are developing a significant research activity in parallel computing, with explicit laboratories implemented in various countries.
- Almost all modern processors integrate several CPU cores. This fact by itself can also be considered as a plain argument for delving into parallel computing concepts. Even end-users of applications that run in parallel could get a better technical picture from being aware of that. Personally, we use to introduce the current fact by considering the OS *multitasking*, which allows to concurrently run several applications even with a single CPU core. Then explain that this virtual concomitance becomes effective with parallel units.
- Energy is a major concern in most of modern activities including those related to the use of computers. It should be recall here that energy is nearly proportional to running time. Thus, reducing the energy consumed by a given software application can be achieved by accelerating its execution. In addition, a processor with clock frequency f consumes more energy than a dual-core with frequency $f/2$ per core. The slogan here is "*parallel computing is energy efficient*".
- Many natural activities are intrinsically parallel. Thus, *natural parallelism* is intuitively expected to be so implemented on computers. Illustrative examples can be taken from *simulation* and *artificial intelligence*.

B. Keep HPC useful for others

This is very important in order to make the HPC pole being perceived as a useful entity and thereby receive rewarding considerations. In addition, keep addressing external requests related to *industrial activities*, *scientific research*, *societal issues*,*services*, and *trainings*, will certainly confirm the importance of the pole and vivify its activities. In practice, it is common to see financial sponsors firmly recommend such connection and also use it a crucial metric for their evaluation and decision. The same exhortation might come from institutional hierarchy or politics. Note that, because of the strong need for HPC availability, some companies implement their own HPC section. Academics could claim permanent investigations, less restricted research focus, and high potential for innovation.

C. Make sure to address local needs

From our personal teaching and collaboration experience, in South America for instance, it clearly appeared that addressing local needs and issues is highly appreciated and valuated. This is normal because otherwise, especially in case of too specific concerns, there is no chance that an appropriate solution will come from elsewhere. In addition, there is a strong desire to have direct hand to the implemented solutions for many reasons including *cost*, *suitability*, *independence*, and *technical maturity*. Thus, an HPC pole should prioritize local issues in their applications portfolio, of course with a certain of level interest in external and prospective scenarios.

D. Be keen in academic collaborations

Standing and acting in an isolated manner is a noticeable factor of stagnation and sort of obscurantism, which is the opposite of what is expected from an academic entity. HPC is moving too fast and changes occur very frequently. Hardware advances and near future projections are regularly announced. New HPC methods, achievements, and challenges are actively made available. Thus, keep collaborating with other actors is essential to survive and to receive consideration. The resulting synergy is another vital benefit to take into account.

III. Implementing an academic HPC pole

This section discusses the main components and objectives of an academic HPC pole.

A. Global organization of the pole

The pole itself is a virtual entity of people with relevant skills. Figure Fig. 2 provides a view of the main units of the pole. Beside this logical clustering, it is important to have the units interacting efficiently. External requests as previously explained are likely to reach the head or research staff. A management policy should (at least roughly) indicate how process such requests.

B. A consistent HPC curriculum

We present and discuss the main basis for a consistent HPC curriculum [28]. Still at the level of an overview, we provide a connected panorama of the key learning units related to *parallel computing*. The teaching part could be complemented with external invited speakers, thus the importance of collaborations, which will also boost the outcome of the research activities.

1) Topics selection: The curriculum should be organized so as to allow coherent inner steps and motivated choices. Figure Fig. 4 shows the main paradigms of *parallel programming*. These paradigms are independent, so the associated classes can be provided without any strong interconnection. However, it should be made possible to follow all of them for those seeking a multi skilled profile, which is technically essential for *hybrid parallel programming*. Regarding how to chose the right way, figure Fig. 5 suggests a a basis for a systematic selection.

2) Applications selection: As previously explained, the selection of the applications to be used as case studies is very important in order to best capture the attention of the audience and also create a strong connection with reality. The effect of parallel computing should convincing. Thus, the main criteria for selecting the target applications are: *not purely fictive*, *good illustrative potential*, *high impact*, *well parallelizable*, *not to complicated to implement in parallel*. For instance, Bioinformatics topics are particularly exciting in the Brazilian context. Indeed, Brazil has a great biodiversity, however knowledge about the genetic resources and the biotechnological potential of Brazilian species is still very limited. Bioinformatics motivations, challenge and perspectives as they are stated by UnB researchers include:

- Great diversity of organisms in Brazil with biotechnological potential
- Major infectious diseases among Brazilian population have been the focus of intensive research
- Number of genome and transcriptome projects is increasing very fast due to Brazilian government support
- Huge amount of data has been generated and has to be processed and analyzed
- Need for active scientific collaborations to provide an efficient computing support to genomic research, with an emphasis on pipeline optimization, information storage and recovering.

We now list some applications extracted from our past and current collaborations:

- One of the main problems Bioinformatics is facing now is how to process genomic data at the rate it is being produced. Once a new biological sequence is discovered, its functional/structural characteristics must be established. For this purpose, newly discovered sequence is compared against the sequences that compose genomic databases, in search of similarities. In order to accelerate the production of results, we could use parallel/distributed variants of the exact methods for pairwise or sequence-profile comparisons, employing many computing units in order to reduce the computation time.
- Flight scheduling is one of the earliest steps in airline operations planning. The schedules quality is crucial for airline profitability and operations reliability. The *fleet assignment* problem consists in deciding which aircraft types (or fleet) will be assigned to the flights. This problem is routinely considered in airline industry, thus the need of a fast computer program to reach good solutions in a reasonable time.
- One of the biggest challenge in Life Science is reading and analyzing the genetic code of the genome. Current high-throughput sequencing machines are able to produce hundreds of millions of short sequences (reads) in a single run and the cost of sequencing has decreased dramatically. However, the task of reconstructing genomes from the large volumes of fragmentary read data still remains an algorithmic challenge that is in practice answered by heuristic and non-satisfactory solutions. Exact methods are computationally costly, thus the need of parallel computing.
- The Amazon has a long history of human settlement. The knowledge that Amazonians have about natural plants and their properties are being lost over time. Currently, there are some attempts to store these information by creating documented catalogs of plants and further analyze them. In this context, number of south America countries that share the Amazonian territory joint together to help Amazonians making their ecosystem safer. The amount of collected information is huge, and process them efficiently is best done by parallel computing.

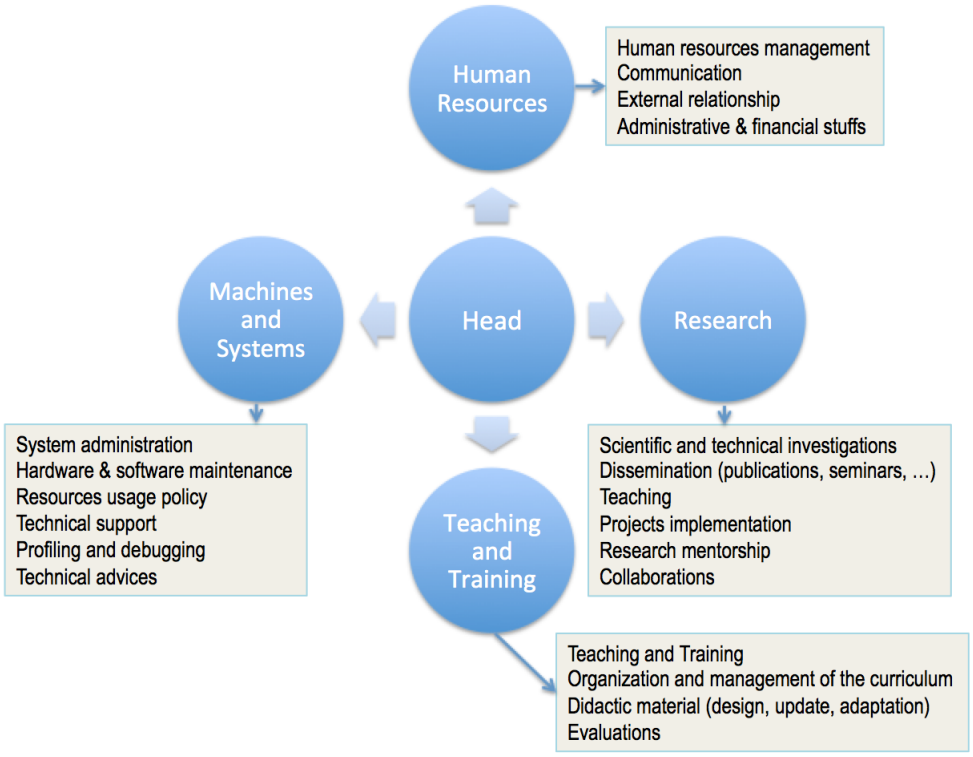

Fig. 2. Overview organization of an academic HPC pole

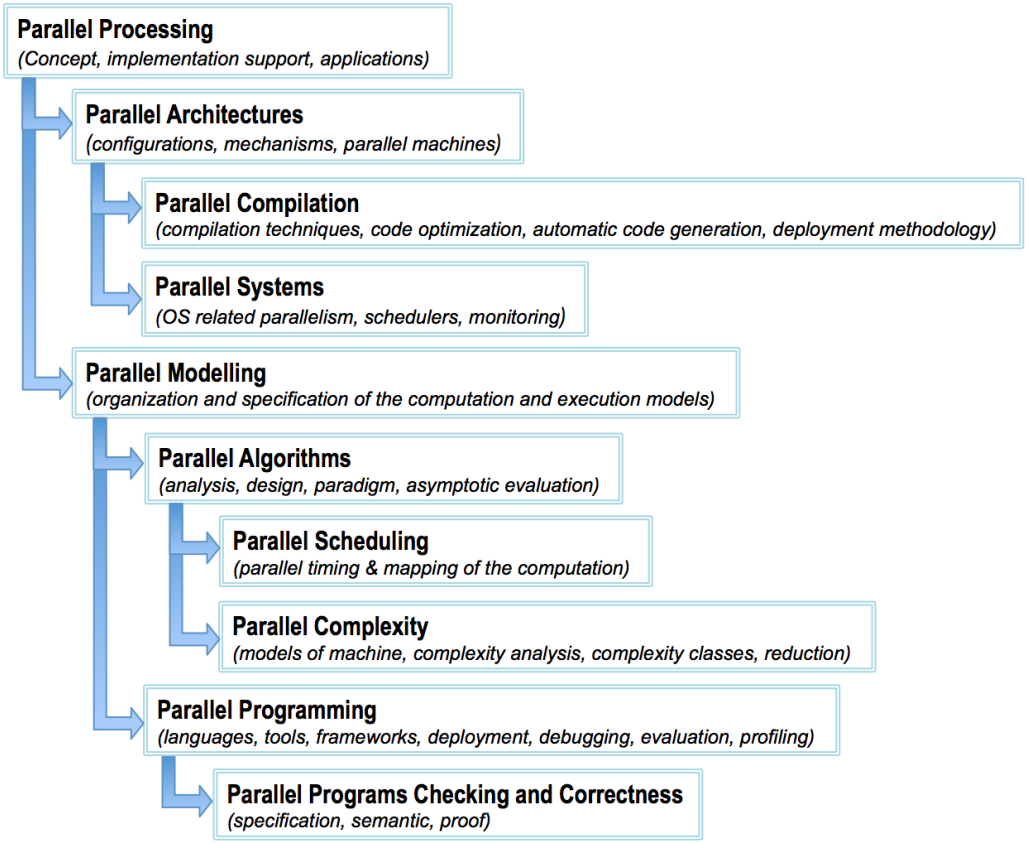

Fig. 3. Dependencies within a parallel computing curriculum

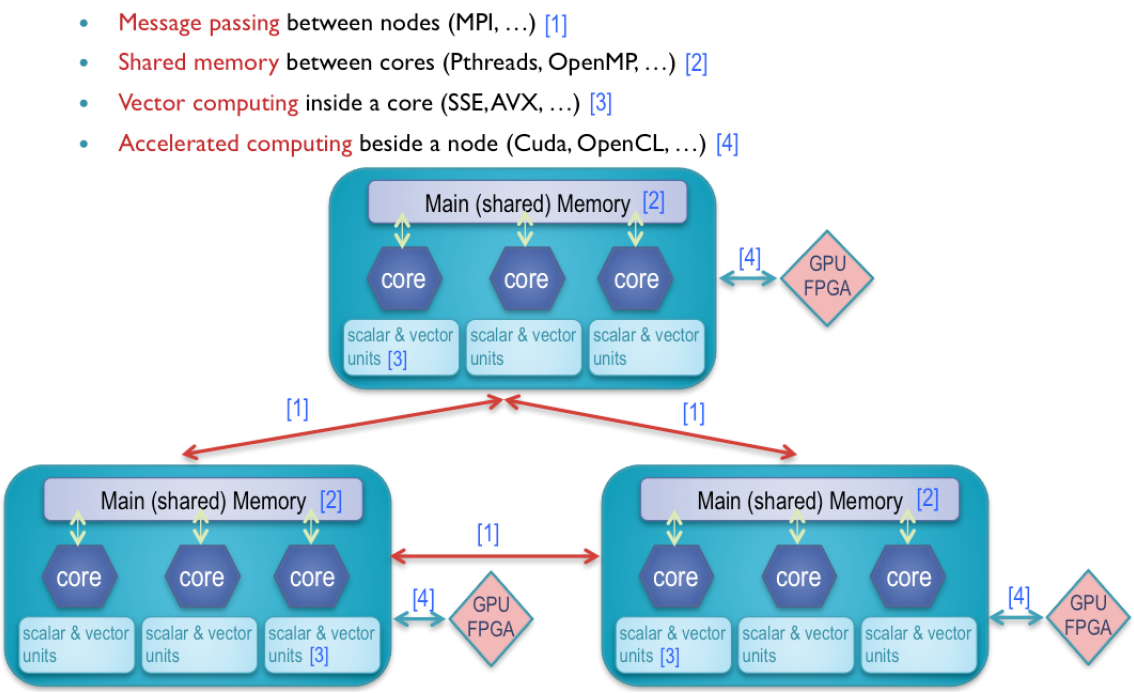

Fig. 4. Main paradigms of parallel programming

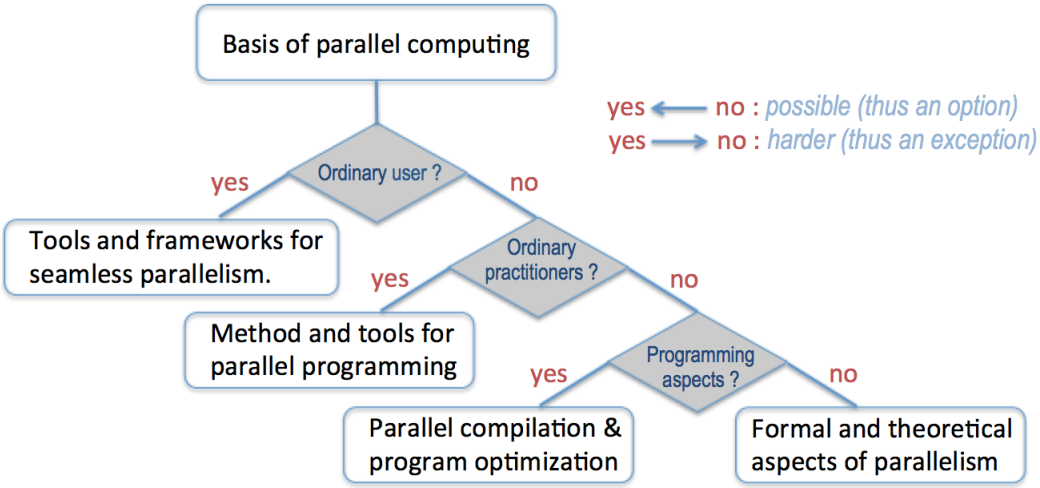

Fig. 5. Flowchart of parallel computing curriculum

IV. Setting up an operational HPC cluster

Any connected machine can be accessed remotely through the network. This is the case for most HPC clusters for which such an access is commonly made available for external users. This is also the mean by which HPC resources are shared between different entities. Nevertheless, there are several reasons for planning a local HPC cluster:

1) fulfilling routine computing requests from local entities;
2) allowing internal or private computing activities;
3) serving as a support for parallel computing practices;
4) acting as a resources center;
5) getting or claiming a computing resources independence;
6) fulfilling the need or reaching a milestone of an HPC project;
7) housing a multi-purpose server;

to name a few. We now describe the way to target this technical achievement and discuss the main items.

The first technical question, maybe implicit, should be "*what is the total peak performance that we want for the supercomputer*?", followed by practical concerns like *design budget*, *maintenance budget*, *lifetime*, *jobs density and frequency*, *cooling requirement*, to name a few. Other secondary questions might focus on *the vendor*, *specific hardware capabilities*, *hardware flexibility*, *robustness*, *suitability to the*

expected usage. From the answers of these questions, a picture of the global configuration can be drawn. Now, let move to technical aspect considerations regarding the goal of building a proprietary HPC cluster.

The first and major concern should be the generic compute node, which could be a *multicore CPU*, a *mono-socket manycore CPU*, or a *multi-socket manycore CPU*, coupled or not with GPU card(s). An example of compute node could be a {mono/multi}-socket Intel Broadwell-EP [31]. Figure Fig.6 displays its main specifications. There pros and cons

# of Cores	12
# of Threads	24
Processor Base Frequency	2.20 GHz
Max Turbo Frequency	2.90 GHz
Cache	30 MB Smart
Bus Speed	9.6 GT/s QPI
# of QPI Links	2
Max Memory Size (depende	1.54 TB
Memory Types	DDR4 1600/1
Max # of Memory Channels	4
Max Memory Bandwidth	76.8 GB/s

Fig. 6. Intel Broadwell-EP main characteristics

for aggregating to many cores within a single node. The main advantage is the one usually considered for hybrid share/distributed memory computing, which is the reduction of the interconnection network. Having less physical nodes is also an acceptable argument. Another one, very important, is energy, both heat and power consumption. Concerning the cons, the situation is like putting all your eggs within the same basket. In case of (even local) failure, it is likely to disable (and loose) the whole node, which thus yields a severe consequence. Another one is at the OS side, user threads (in a multithreaded program) will collude with standard processes. From the sustained performance viewpoint, a heavy concurrency between too many threads threads within the same node might severely affect the program scalability. This bottleneck mainly results from penalizing memory accesses, which is a crucial concern with NUMA configurations.

Next, since the available compute nodes will have to communicate to each other, hence come questions related to the network like *speed (bandwidth and latency)*, *topology*, and *reliability*. Considering a simple Ethernet network will be sufficient and operational.

Storage capacity should be evaluated and extended if necessary. Also for this item, backup mechanisms should be addressed.

For software aspects, the choice will focus on *operating system*, *compilers*, *programming and runtime libraries*, *debug tools*, *monitoring tools*, and *(batch) schedulers*.

For a technically detailed step-by-step explanation and tutorial to build up a standard HPC cluster, we suggest [32].

V. Conclusion

Designing and implementing a parallel computing curriculum has became a common goal. Indeed, the topic of *parallel and distributed computing* is now vital because of the emergence and pervasiveness of parallel systems. This topic has several directions and a wide range of applications, thus should be considered a good level of conscience. Managing all related activities within an HPC pole is clearly a good idea, which needs to be considered methodologically. The enthusiasm behind is motivated by the active evolution of HPC systems and the increasing demand of efficient computing solutions for numerous applications. Important facts, advices, schematic views, recommendations and descriptions are reported in this paper in the perspective of creating an active academic HPC pole.

References

[1] R. Buyya , *High Performance Cluster Computing: Architectures and Systems* (volume 1 volume 2), Prentice Hall, 1999.

[2] R. Correa, I. de Castro Dutra, M. Fiallos, L. F. Gomes da Silva (Eds) , *Models for Parallel and Distributed Computation: Theory, Algorithmic Techniques and Applications*, 2nd ed. Springer, December 10, 2010.

[3] B. Chapman, G. Jost, R. van van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming*, The MIT Press, October 12, 2007.

[4] J. Dongarra, I. Foster, G. C. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White (Eds), *The Sourcebook of Parallel Computing* , 1st ed. Morgan Kaufmann, November 25, 2002.

[5] J. Dongarra and P. Luszczek, Introduction to the HPC Challenge Benchmark Suite, tech. report, Univ. Tennessee, 2004.

[6] F. Gebali, *Algorithms and Parallel Computing* , Wiley, April 19, 2011.

[7] J. Hromkovic, *Communication Complexity and Parallel Computing* , Springer, December 1, 2010.

[8] J. JaJa, *Introduction to Parallel Algorithms* , 1st ed. Addison-Wesley Professional, April 3, 1992.

[9] A. Grama, G. Karypis, V. Kumar, A. Gupta, *Introduction to Parallel Computing (2nd Edition)* , 2nd ed. Addison-Wesley, 2003.

[10] R. Greenlaw, H. J. Hoover, W. L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory* , Oxford University Press, USA, April 6, 1995.

[11] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*, 1st ed. CRC Press, July 2, 2010.

[12] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming* , 1st ed. Morgan Kaufmann, March 14, 2008.

[13] M. Hill and M. Marty, *Amdahls law in the multicore era*, Computer, vol. 41, no. 7, pp. 33 -38, 2008.

[14] G. E. Karniadakis, R. M. Kirby II, *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation* , Cambridge University Press, June 16, 2003.

[15] D. B. Kirk and W. W. Hwu , *Programming Massively Parallel Processors: A Hands-on Approach* , Morgan Kaufmann, February 5, 2010.

[16] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* , Morgan Kaufmann Pub, September 1991.

[17] F. T. Leighton, *Introduction to Parallel Algorithm and Architectures: Arrays, Trees, and Hypercubes*, Morgan Kaufmann,2nd ed. San Mateo CA, 1991.

[18] T. G. Lewis and H. El-Rewini, *Introduction to Parallel Computing*,2nd ed. Prentice-Hall, Englewood Cliffs, USA, 1992.

[19] Mano, *Computer System Architecture*, ISBN-10: 8131700704, , 2nd ed. Pearson Education India, 2007.

[20] P. Pacheco, *An Introduction to Parallel Programming* , 2nd ed. Morgan Kaufmann, 2011.

[21] D. Patterson, *Computer Architecture: A Quantitative Approach Fifth Edition* , 2nd ed. Morgan Kaufmann, 2011.

[22] T. Rauber and G. Rnger, *Parallel Programming for Multicore and Cluster Systems* , 1st ed. Springer, March 10, 2010.

[23] P. Pacheco, *Parallel Programming with MPI* , 1st ed. Morgan Kaufmann, October 15, 1996.

[24] M. J. Quinn, *Parallel Programming in C with MPI and OpenMP* , McGraw-Hill Education , January 2008.
[25] S. H. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation* , 1st ed. Springe, December 10, 1999.
[26] C. Severance and K. Dowd, *High Performance Computing*, online book, Rice University, Houston, Texas, 2012.
[27] L. R. Scott, T. Clark, B. Bagheri, *Scientific Parallel Computing* , Princeton University Press, March 28, 2005.
[28] C. Tadonki, *Basic parallel and distributed computing curriculum*, Second NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar'12), in conjunction with the 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS), May 2012, Shanghai, China
[29] C. Tadonki, *High Performance Computing as Combination of Machines and Methods and Programming* , HDR defense, University Paris-Sud, Orsay, May 16, 2013.
[30] Nagel, Wolfgang E.; Krener, Dietmar B.; Resch, Michael M (Eds.), *High Performance Computing in Science and Engineering '12*, Springer Book Archives,, 2013.
[31] https://ark.intel.com/products/91767/Intel-Xeon-Processor-E5-2650-v4-30M-Cache-2_20-GHz
[32] http://www.wikihow.com/Build-a-Supercomputer