EasyChair Preprint
№ 4000

HS-AUTOFIT: a Highly Scalable AUTOFIT Application for Cloud and HPC Environments

Antonio Corradi, Giuseppe Di Modica, Luca Evangelisti,
Anna Fiorini, Luca Foschini and Luca Zerbini

August 2, 2020

HS-AUTOFIT: a highly scalable AUTOFIT application for Cloud and HPC environments

Antonio Corradi, Giuseppe Di Modica, Luca Evangelisti, Anna Fiorini, Luca Foschini, Luca Zerbini
Department of Computer Science and Engineering
University of Bologna, Italy
Email: {firstname.lastname}@unibo.it

Abstract—Technological progress is leading to an increase of instrument sensitivity in the field of rotational spectroscopy. A direct consequence of such a progress is the increasing amount of data produced by instruments, for which the currently available analysis software is becoming limited and inadequate. In order improve data analysis performance, parallel computing techniques and distributed computing technologies like Cloud or High Performance Computing (HPC) can be exploited. Despite the availability of computer resources, neither Cloud computing nor HPC have been fully investigated for identifying unknown target spectra in rotational spectrum. This paper proposes the design and implementation of a Highly Scalable AUTOFIT (HS-AUTOFIT), an enhanced version of a fitting tool for broadband rotational spectra that is capable of exploiting the resources offered by multiple computing nodes. With respect to the old program version, the new one scales on multiple computing nodes thus guaranteeing higher accuracy of the fit function and consistent boost of execution time. The result of tests conducted in real Cloud and HPC environments demonstrate that HS-AUTOFIT is a viable solution for the analysis of huge amount of data in the addressed scientific field.

I. INTRODUCTION

Technological progress is increasingly improving data collection and analysis in many scientific fields. Particularly, the recent development of high-speed digital electronics provides new results from chirp pulsed Fourier transform microwave (CP-FTMW) spectrometers, thus producing broadband rotational spectroscopy measurements of molecular clusters with unprecedented sensitivity. Much better sensitivity means denser spectra and more precision in target spectra identification, but also huge amount of data to analyze [1].

Despite the high density of lines in spectra and the problem of analyzing them, high computing power technologies, like Cloud and High-Performance Computing (HPC), have not yet been fully investigated by researchers in the field of quantum chemistry. With these systems, it is possible to process data through distributed computing, improving efficiency and maximizing performance of data analysis. Many applications originally developed to run on personal computers can leverage distributed computing. The improvement in performance is bound to hardware availability, such as number of computing nodes, number of CPUs, CPU frequency, number of CPU cores, memory and so on. So, the switch from a a single node system to a scalable system with conceptually unlimited resources will greatly increase the available computing power.

This research focuses on improvement of an automated fitting tool, namely AUTOFIT [2], for the identification of unknown target spectra in a broadband rotational spectrum.

The program takes as input the rotational constants A, B and C, a set of dipole moments predetermined thanks to quantum chemistry and some experimental parameters like a frequency interval, limits on the intensity of the transitions and the rotation temperature of the analyzed molecular sample. From these data, AUTOFIT will search an input microwave data set for an experimental spectrum consistent with the predicted parameters. The data exploration process is committed to multiple "workers" (computing threads) that are assigned portions of the whole input data to explore. Trreads are able to do their work independently of each other and run in a parallel fashion.

Firstly, we analyze the structure of AUTOFIT algorithm, and in particular, the branch that enforces parallel computation to speed up the performance. Secondly, we propose a new version of the algorithm, named Highly Scalable AUTOFIT (HS-AUTOFIT), that is designed to fit distributed computing environments. Finally, to show the potential of HS-AUTOFIT, we present the result of experiments run on both Cloud and HPC environments. Results prove that the enhanced algorithm is capable of scaling in both environments, though different performance gains were observed.

The rest of the paper is structured in the following way. In Section II, we briefly recall the basic principles of the distributed and parallel programming, as well as provide an insight on AUTOFIT and the issues addresses in the field of rotational spectroscopy. In Section III, we analyze AUTOFIT algorithm and propose a modification that will enable it to scale on distributed computing environments. In Section IV, we discuss some tests carried out to evaluate the performance gain of the enhanced algorithm both in Cloud and in HPC environments. Finally, Section V concludes the work.

II. BACKGROUND AND RELATED WORK

A. AUTOFIT application

Rotational spectroscopy (or *microwave spectroscopy*) concerns measurement of energy transitions between quantized rotational states of molecules in the gas phase. In rotational spectroscopy, molecules are classified following their symmetry in spherical, linear and symmetric rotors. For these rotators, analytical expressions can be derived that describe the energetic terms. Rotational levels are theoretically obtained by considering molecules as rigid rotators and subsequently applying corrective terms. The comparison between spectra and theoretical expressions provides numerical values of angular moments of inertia from which it is possible, in favorable

cases, to derive accurate values of angles and bond lengths of molecules.

Thanks to the development of high-speed digital electronics, a new tool for identification of rotational spectrum called Chirped-Pulse Fourier Transform Microwave Spectroscopy (CP-FTMW) has emerged. This spectroscopy produces a significant amount of data, increasing the sensitivity of the instrumentation. This increase has in fact produced spectra with a very high density of molecular transitions that can go up to 1 Mhz. With such a high band density, manual fitting of spectra is really inefficient since the models associated with the different combinations of rotational transitions are difficult to identify. Consequently, new solutions need to be devised. An approximation of the rotational spectrum can be obtained by fitting at least three transitions in order to calculate the rotational constants A, B and C. The validity of the predicted spectrum can be verified by predicting other rotational transitions and checking their presence in the experimental spectrum.

In order to analyze the complex spectra from these broadband measurements, an automated spectral assignment program called AUTOFIT was developed [2]. AUTOFIT was originally conceived by Prof. Brooks H. Pate at the University of Virginia (USA) and subsequently developed by Prof. Steve Shipman of College of Florida (USA) and his collaborators. The program implements an algorithm that searches for all possible combinations of three transitions in a given frequency range. This algorithm is able to determine the rotational constants even without the use of the initial spectroscopic parameters derived from computational chemical software (example: ab initio [3] or density functional theory (DFT) [4] calculations). The spectrum is calculated using both predicted intensity and frequency values of rotational transitions. The program takes as input the rotational constants A, B and C, a set of dipole moments dipole predetermined thanks to quantum chemistry and some experimental parameters like a frequency interval, limits on the intensity of the transitions and the rotation temperature of the analyzed molecular sample. From to these data, AUTOFIT will search in the set of rotational transitions present in the experimental spectrum consistent with the aforementioned parameters.

AUTOFIT operates as follows. First, it generates an initial forecast from the parameters provided in the input file. AUTOFIT is able to independently choose three transitions on which to perform the fitting; however it also provides the user with the option to choose the three transitions manually. AUTOFIT then performs the fitting on all possible combinations of transitions of the spectrum within the three search windows (frequency range) corresponding to the three transitions.

From an algorithmic point of view, the detection of a molecule within an experimental spectrum can be implemented as an exploration of many combinations of triplets. The number of algorithm steps necessary to the detection purpose varies according to the accuracy of the goal we need to attain. Specifically, that number depends on the size of the frequency window in which to seek for the transitions, therefore it is not rare that a few thousands or even millions of combinations have to be checked out. In its original design, AUTOFIT can analyze 35-50 triplets per second per core, therefore a typical 4-core processor can approximately process 250 triplets per second. A typical run takes from 2 to 3 hours considering a frequency window of 200-300 MHz. The increase in the window search is sometimes necessary because often predictions are not as accurate as requested (the experimental results go far from the theoretical predictions). However, enlarging the window search size can cause unacceptable growths of the execution time, thus making the tool useless.

Therefore, the challenge addressed in this paper is to elaborate a new algorithmic approach that can accommodate any level of accuracy requested and, at the same time, can provide much better execution times.

B. Distributed and Parallel computing

According to an authoritative definition, *"Distributed computing deals with all forms of computing, information access, and information exchange across multiple processing platforms connected by computer networks."* [5]. The distinguishing feature of distributed computing is the use of *multiple processing platforms*, connected by network links, on which a number of software programs are executed and collaborate with each other to a given purpose. Distributed computing breakthrough is dated back to late 1970. Despite the advantages are well recognized by researchers, it also comes with many issues that are under investigation by several research communities ever since. Discussing about advantages and issued of distributed computing is however out of the scope of this work.

Parallel computing can be recognized under the big umbrella of distributed computing. As a general definition, parallel computing is a type of distributed computation in which many calculations or the execution of processes are carried out simultaneously [6]. There exists many parallel programming models. The one that fits the purpose of this paper inspires to what is usually referred to as data parallelism [7]. It refers to the process of decomposing an application into multiple tasks, assigning them a portion of the application's workload, running the tasks in parallel (i.e., at the same time) on a number of computing resources, collecting and eventually assembling the tasks output. The expected result of running an application in a parallel fashion is a speed-up of the application execution time in comparison to running the same application on a single computing resource.

High Performance Comuputing (HPC) is a technology used in cluster computer contexts to create processing systems capable of providing very high performance (in the order of PetaFLOPS) through the use of parallel calculation. HPC clusters are usually equipped with a high number of processors, large memory capacity and a high-bandwidth and low-latency interconnections. HPC is widespread in scientific environments, where huge amounts of data have to be analyzed by scientific application. HPC offers scientists a way to easily scale up their application and carry out quick data analysis via parallelization of the calculus over the cluster nodes. Usually, cluster nodes communicates via a lightweight protocol called Message Passing Interface (MPI). MPI is a standard specification for communication between nodes belonging to clusters of computers running a parallel program. MPI mainly addresses the parallel programming model of message transmission, in which data is moved from the address space of one process to that of another through cooperative operations on both processes.

Finally, in the panorama of distributed computing technologies, *Cloud Computing* has delivered the promise of addressing computing needs in a flexible and dynamic way. Cloud Computing manages to provide computing resources to requesting users in the form of a public utility such as water and electricity [8]. Cloud computing offers data center owners a virtualization paradigm that allows them to address the need of requesting users in a very accurate way and to optimize the resource utilization. Computing resources in a Cloud environment are offered in the form of virtual processors/cores, virtual memory and virtual storage, thus forming a *virtual node* on top of which the user can install any operating system. Communication among virtual nodes is implemented by Virtual LAN (VLAN). Processes distributed over multiple cloud nodes may then use sockets to interface to each other. With respect to the HPI, which implements a rigid architecture of very powerful and homogeneous computing nodes, Cloud can offer computation in a more flexible way.

III. DESIGN AND IMPLEMENTATION OF A HIGHLY SCALABLE AUTOFIT APPLICATION

The main flaw of AUTOFIT is that it was designed to run on a single machine. Usually, off-the-shelf Desktop PCs are used to this purpose. Still, had more powerful machines been employed, there would not be guarantee that the requested accuracy can be meet. In other words, *vertically* scaling is not the answer to the problem stated in Section II-A.

Instead of counting on powerful single machine (from now on, computing node), we propose to exploit the *horizontal* scaling, i.e., the possibility of adding more nodes whenever requested, and enforce a simultaneous, parallel run of the algorithm on all nodes. Yet, to fully exploit the parallel computing potential, AUTOFIT's naive algorithm must be carefully re-designed. The revised algorithm will have to split the application workload into smaller units and distribute them to as many "workers" capable of running in parallel on multiple computing nodes. To this end, it is essential to understand the inter-dependency of data computed by different workers and deal with possible issues concerning with the synchronization and inter-communication.

The flow diagram of AUTOFIT algorithm is depicted in the Figure1. By design, AUTOFIT does enforce parallel computation by exploiting the Cores of the node it runs onto (we can refer to it as "local parallelism"). Let us focus on the branch of the algorithm that enforces local parallelism.

The algorithm assumes that N Cores are available on the Desktop PC. As the reader may notice, the triples generation step splits up the list of triples into N equally sized sub-lists that are, in their turn, assigned to as many computing tasks. Each task, exploiting the computing power of one Core, runs the fit function against every triple contained in its own sub-list. At the end of this step, every triple is marked with a *Score* and the "triple-score" pair is appended to an output text file. The last algorithm step gathers all tasks' output files, merges the "triple-score" pairs, sorts them by the score and produce one final *Output File*.

We remark that the accuracy of the final result can be improved by increasing the size of the input search window, which, in turn, increases the number of combinations of
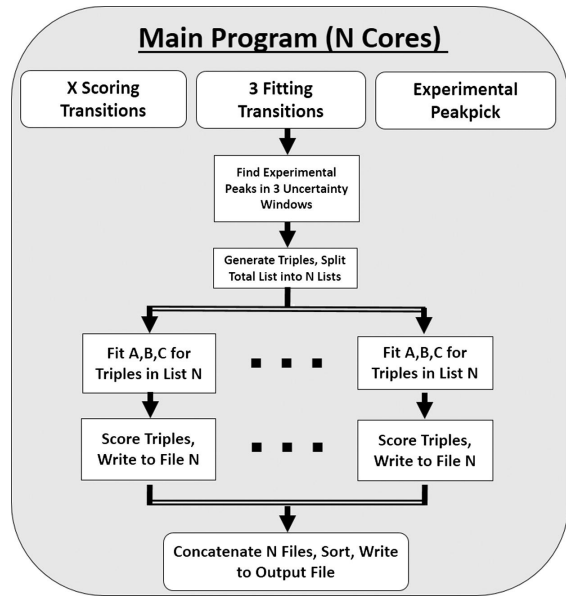

Fig. 1. Flow diagram of AUTOFIT

transitions (triples) to fit. The longer the input triples list, the heavier the workload each task is assigned. To alleviate the per task workload, a viable yet trivial solution might be to increase the number of tasks. Unfortunately, had the tasks number to exceed that of the available Cores the benefit achievable in terms of run-time speed would be less than the expected.

In this paper, we propose Highly Scalable AUTOFIT (HS-AUTOFIT), a modified version of the AUTOFIT algorithm capable of scaling to computing nodes other than to local node's Cores. Same as with the Cores, the scalability limit of the new algorithm is imposed by the number of computing units available for computation. Thanks to consolidated technologies such as HPC and Cloud, adding computing nodes on demand is undoubtedly an easy and viable solution.

Adding nodes to computation means to have more power available to fit triples in a shorter time. Of course, the join of nodes must be carefully coordinated since services like inter-node communication and/or sharing of resources needs to be provided. To this end, we will call on the well known distributed computing paradigm *Master-Slave*. In the prospected context, the *Master* is committed to receiving the application workload, distributing it to *Slaves*, receiving and merging the output of the Slaves computation. Zooming in the triples fitting step of the process, each Slave will still be able to exploit its own Cores and recursively apply parallel computing by distributing its workload to Cores.

In this specific case study, we decided to assign to the Master the twofold role of coordinator of distributed computing operations and worker (Slave) as well. We also embedded all data entry operations and the loading of experimental spectra into an "Input" component. The Input component produces files containing all the triples to fit, and also provides the specifications to fit them on a predicted geometry and experimental parameters. The Master-Slave architecture of HS-AUTOFIT is depicted in Figure 2.

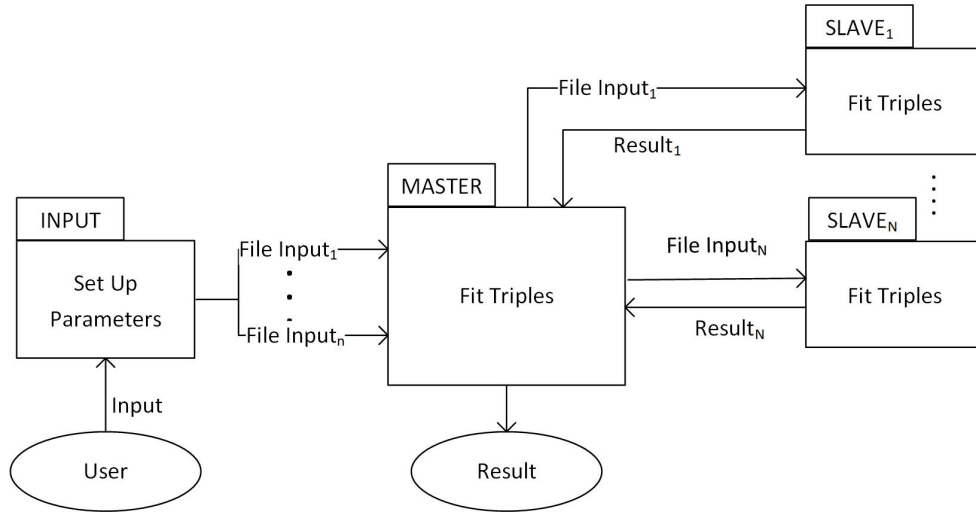Based on this architecture, we implemented two versions

Fig. 2. The HS-AUTOFIT master-slave architecture

of the HS-AUTOFIT program that run on Cloud and HPC environments respectively. The two versions mainly differ to each other on the way communication between the Master and the Slaves is implemented. Both communication implementations, though, offer the following services: transfer of generic data type and, in general, small sized data (e.g., configuration parameters); transfer/sharing of text files (those containing the triples to fit); support for synchronized communication mode. Specifically, we used the *Socket* interfaces to implement Master-to-Slave communication in Cloud environments, while opted for the *Message Passing Interface (MPI)* protocol to implement the communication among nodes in HPC environments. Both Sockets and MPI are supported in Python 2.7, the language used to code the naive version of AUTOFIT on which we applied the aforementioned modification.

IV. HS-AUTOFIT PERFORMANCE EVALUATION

To evaluate the performance of HS-AUTOFIT, we ran several tests in both physical and virtualized systems. Every node involved in the test was equipped with Python and all necessary libraries, as well as the fundamental tools needed by HS-AUTOFIT to execute.

The chosen performance indicator is time HS-AUTOFIT takes to fit all triples provided in input. Also, different workloads were tested by tuning various size of the search window, which, as mentioned earlier, produced different number of combinations of transitions to fit.

Tests were run on a physical node, on Cloud nodes and on HPC nodes respectively. For what concerns the experiment on a physical node, we chose a Desktop PC Intel Core i7-4700MQ 2.40 GHz Quad-Core, 12 GB RAM. As for the test bed on the Cloud, we opted for Amazon Web Services (AWS) [9], that offer on demand compute and storage services. We tested HS-AUTOFIT on five EC2 c5d.xlarge instances, each supplied with 4 vCPU, 8 GB Memory, a clock speed of up to 3.5 GHz, 20 GB Storage (EBS). Instances were connected to each other through a 10 Gigabit virtual network. Finally, HPC experiments were conducted on two different clusters hosted

TABLE I.   EXECUTION TIMES OBSERVED IN THE CASE OF ONE THREAD

	Number of input triples to AUTOFIT			
	7600 (50 Mhz)	34848 (100 Mhz)	253456 (200 Mhz)	617136 (300 Mhz)
Execution time	1m 20s	7m 20s	50m 30s	2h 4m

by CINECA [10], whose details are disclosed later on in the paper.

We highlight that all tests run on physical node, Cloud and HPC nodes were repeated several times and obtained results were averaged.

A. Test on a physical node

This test is aimed at showing that the scalability of the naive AUTOFIT application is limited. We measured the execution time of AUTOFIT for 4 different search windows (50 Mhz, 100 Mhz, 200 Mhz, 300 Mhz), which produced inputs to the program of 7600, 34848, 253456 and 617136 triples respectively.

In Table I we reported AUTOFIT execution times observed in the single-threaded case. As expected, the higher the number of triples to fit, the longer the execution time. Afterwards, we tested the multi-threaded cases. Specifically, we repeated the experiment with 2, 4 and 8 threads respectively. Figure 3 shows that the use of multiple threads generally improves the performance with respect to the single-threaded case, but tends to become less relevant in the case of 8 threads. If we tried to go further than 8 threads, we would get very poor performance.

B. Test on Cloud

For this specific test we just report the case of the highest number of triples (617136). We point out that for each node, the application is configured to create 4 threads. We wanted to measure the performance gain as new nodes are added to the computation. In the case of 1 node the naive AUTOFIT program was run, while for 2,3, and 5 nodes HS-AUTOFIT was employed. Test results are reported in Figure 4. In the case
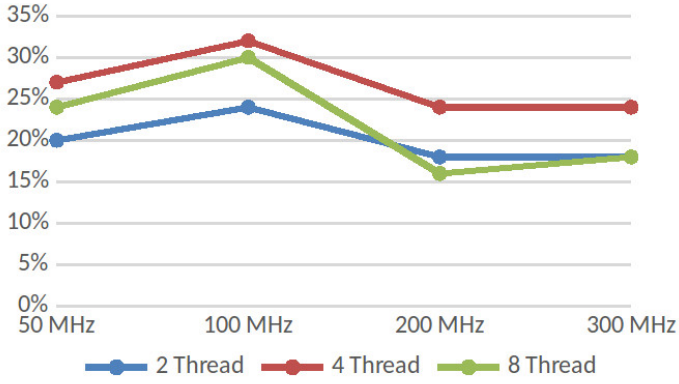
Fig. 3. Performance improvement when multi-thread is employed

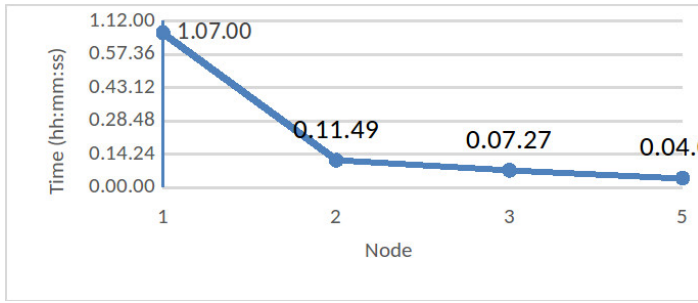of 2,3 and 5 nodes, we observed an improvement of 68%, 69%, and 89% respectively.

Fig. 4. Performance gain when cloud nodes are added to computation

C. Test on HPC

In HPC clusters, nodes are considered as processes of the entire supercomputer. They are identified by a *rank* that can be used for communicate with other nodes/processes. Storage is designed as a parallel and distributed filesystem, optimized for infrequent access to large amounts of data. Figure 5 shows a layered scheme of a HPC cluster similar to those used by CINECA.

We remind that HS-AUTOFIT follows the Single Program Multiple Data (SPMD) approach, so that each dedicated node will perform the same operation (triple fitting) on a given portion of the data, which is passed to it by means of a file created by the Input component of the program. Communication between the various nodes/processes in the HPC is limited to sending the data from the Master to the various Slaves and back, via the MPI protocol.

HS-AUTOFIT was tested on two different cluster systems hosted at CINECA. The objective was to evaluate the performance achievable by HS-AUTOFIT in HPC clusters with respect to that attained in the Cloud environment. To this end, same HS-AUTOFIT input parameters of the Cloud test were used.

In the following we discuss some details of the two systems and present the results of their respective tests.

TABLE II.   MAIN FEATURES OF GALILEO HPC CLUSTER

Model	IBM NeXtScale
Architecture	Linux Infiniband Cluster
Nodes	360 Intel Broadwell 40 nVidia K80 GPU
Processors	2 x 18-cores Intel Xeon ES-2697 2.30 Ghz 2 x 8-cores Inter Haswell 2.40 Ghz 2 NVIDIA K80 GPUs
Network	Intel QDR (40Gb/s) Infiniband switches
Performance	1 TFLOPs
Memory	2 TB
RAM	128 GB/node, 8 GB/core

TABLE III.   MAIN FEATURES OF MARCONI A2 HPC CLUSTER

Model	Intel Server Board
Architecture	Intel Omni Path Cluster
Nodes	3600
Processors	Intel Knights Landing (KNL) 1 x Xeon Phi 7250 (KNL) at 1.4 GHz
Cores	68 cores/node = 244800
Network	Intel OmniPath Edge Switch 100
Performance	10.8 PFlop/s
Memory	17 PB
RAM	96 GB/node
Power	1300 kW

1) GALILEO cluster.: GALILEO is a hybrid HPC Cluster. Out of its 848 nodes, 768 are equipped with Intel Phi 712p accelerator, 80 with NVIDIA K80. Table II shows the main technical characteristics of GALILEO cluster.

For the sake of brevity, we present the result of tests conducted on a configuration of HS-AUTOFIT parameters that generated 617136 input triples to fit. In Figure 6 we report the execution times of a number of program executions over different computing configurations.

Although execution times get shorter with the increase of dedicated computing power, the equivalent configuration run on the Cloud managed to outperform the HPC's. In fact, if we focus on the 5 nodes/4 threads configuration, the 4 minutes and 4 seconds recorded in AWS test is far better than the 8 minutes and 29 seconds observed in GALILEO (twice the execution time). After a careful analysis of GALILEO's architecture, we came to the conclusion that the cause of the slowdown could lie in frequent reading and writing accesses to the file system. In fact, as mentioned before, GALILEO uses a distributed and parallel files system storage solution which is optimized for supporting few accesses to large amounts of data. HS-AUTOFIT is based mainly on the exchange of data via text files: threads read the input files assigned to them and writes output files which, in turn, are reworked to obtain the final result file.

2) MARCONI cluster: MARCONI A2 Cluster offers the scientific community a energy efficient computing facility. The cluster takes advantage of the new Intel Omni-Path architecture, which guarantees high performance interconnections to allow efficient scaling of thousands of servers in the system. One of the objective pursued by the project that sustains the cluster is to gradually increase the the computational power up to 50 Pflop/s without ever exceeding the limit of 3MW of electrical absorption. Table II shows the main technical characteristics of MARCONI cluster.

For MARCONI Cluster, we present the results of tests run on input of 617136 and 768586 triples respectively. In Figure 7 execution times of two computing configurations per the two
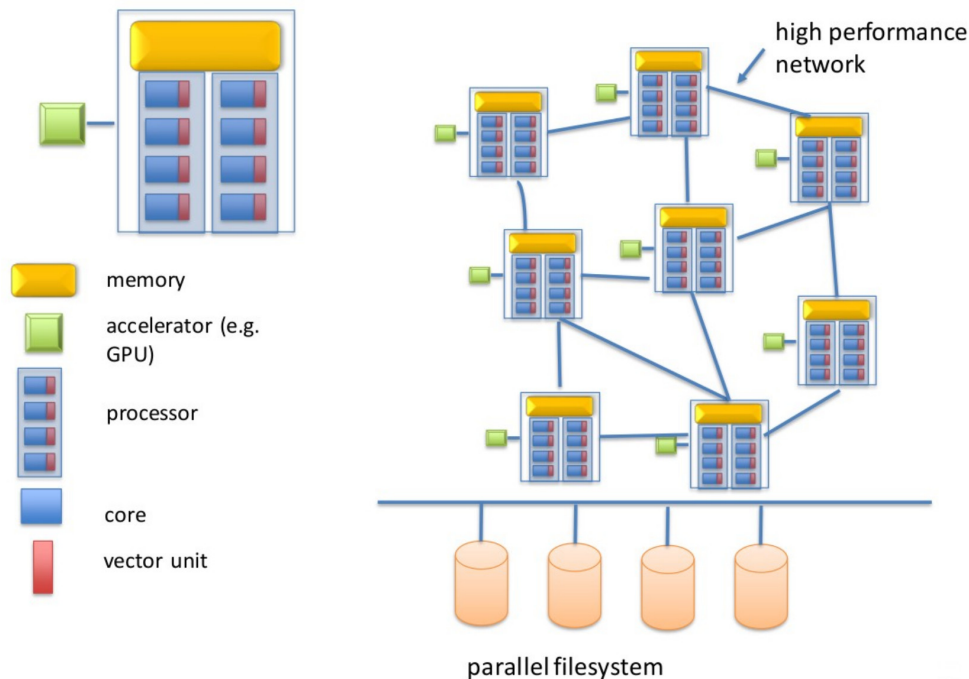
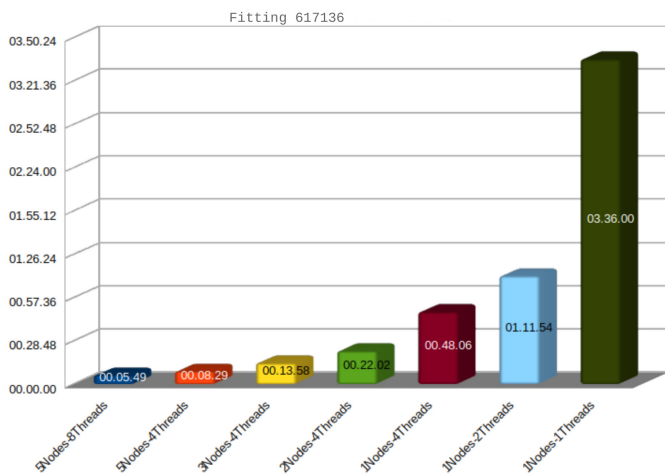Fig. 5. Scheme of an HPC cluster at CINECA

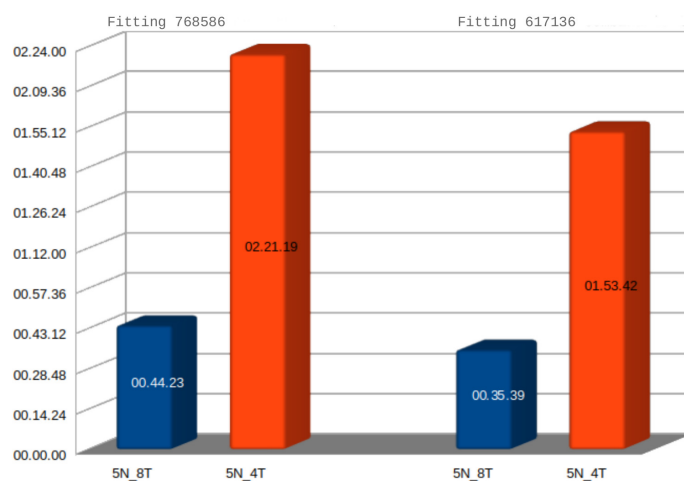Fig. 6. HS-AUTOFIT execution times on GALILEO cluster

Fig. 7. HS-AUTOFIT execution times on MARCONI cluster

inputs are depicted.

Again, looking at the 617136 triples case, execution times of HS-AUTOFIT on MARCONI A2 are worse than those recorded on GALILEO and AWS.

D. Final considerations

The test results proved that, in order to boost the performance of a scientific application, several aspects need to be taken into account. For sure, a good application design is a nice starting point. Yet, the design must not neglect the architecture of the parallel computing system that the application will run onto. Some applications may happen reach very good performance on a cluster that, instead, turns out to be totally inefficient for other applications. For instance, I/O bounded applications, for instance, are totally unfit to run on clusters whose storage are optimized for rare accesses to very large files. On the other end, it is well known that an excessive use of message passing in clusters may lead to poor performance, due to the high latency introduced by the synchronous communication of MPI. The rule of thumb could be making a deep analysis of the application algorithm and, accordingly, finding a correct balance of the two approaches.

V. CONCLUSION AND FUTURE WORK

Broadband chirped-pulse Fourier transform microwave (CP-FTMW) spectrometers have increased the sensitivity for molecular rotational spectroscopy. The higher sensitivity in

broadband rotational spectroscopy measurements of molecular clusters produces huge amounts of data. Scientific applications exploiting the computing power of a single machine are unfit to handle and analyze this data. In this paper, we have first examined AUTOFIT, a fitting tool for rotational spectra that the scientific community has used for identifying unknown target spectra in rotational spectrum. Then, we have presented HS-AUTOFIT, an enhanced version of AUTOFIT that exploits a parallel computing technique to scale on multiple computing resources, be them virtual resources offered by a Cloud provider or nodes of a HPC cluster. Experimental results showed that HS-AUTOFIT guarantees much shorter execution times in both computing environments. Future research will focus on better analysing the performance obtained in MARCONI testbed, and laying down guidelines that help application designers get the best out of high performance computing contexts.

REFERENCES

[1] N. A. Seifert, I. A. Finneran, C. Perez, D. P. Zaleski, J. L. Neill, A. L. Steber, R. D. Suenram, A. Lesarri, S. T. Shipman, and B. H. Pate, "Autofit, an automated fitting tool for broadband rotational spectra, and applications to 1-hexanal," *Journal of Molecular Spectroscopy*, vol. 312, pp. 13 – 21, 2015.

[2] U. of Virginia, "Autofit, an automated triples fitting program for broadband rotational spectroscopy," https://faculty.virginia.edu/.archived/bpate-lab/autofit/intro.html, 2013.

[3] Q-chem, "Q-chem 5.2: Facilitating worldwide scientific breakthroughs," https://www.q-chem.com/, 2020.

[4] Cp2K, "Open source molecular dynamics," https://www.cp2k.org/, 2020.

[5] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, 2008.

[6] G. S. Almasi and A. Gottlieb, *Highly Parallel Computing*. USA: Benjamin-Cummings Publishing Co., Inc., 1989.

[7] J. O'Donnell, *Data Parallelism*. London: Springer London, 1999, pp. 191–206.

[8] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in *10th IEEE International Conference on High Performance Computing and Communications (HPCC'08)*, Sep. 2008, pp. 5 –13.

[9] Amazon, "Amazon web services," https://aws.amazon.com/, 2020.

[10] Cineca, "Consorzio interuniversitario per il calcolo automatico dell'italia nord orientale," https://www.cineca.it/, 2020.