Characterizing Grid experiments in Bioinformatics for an efficient scheduling

Ignacio Blanquer[1], Abel Antonio Carrión[1], Vicente Hernández[1]

[1] Instituto Universitario de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, Universidad Politécnica de Valencia,
46022 Valencia, Spain
{iblanque, acarrion, vhernand}@itaca.upv.es

Abstract. Bioinformatics is an area which involves the execution of many computing-intensive applications. Due to the development of new complex techniques and the increasing size of databases, this area demands the execution of experiments which exceed the resources of most research groups. The use case presented in this work is one of the most representative applications in the Bioinformatics field: computing the alignment of genomic and proteomic samples with respect to annotated databases through BLAST, developed at the NCBI in USA. Performing the homology search of one sequence with BLAST, even with large databases such as GenBank, takes only a few minutes. However, processing millions of sequences, with a sequential approach, requires years of CPU computation. Nevertheless, the computing time of this massive parallel application can be intensively reduced by means of e-Science infrastructures, such as EGEE, EELA and some NGIs, splitting the work into thousands of loosely-coupled tasks. In order to improve the performance of these experiments, a key aspect has proven to be the development of sophisticated automatisms which predict the jobs' elapsed time. Thus, this article focuses on describing all the details regarding the characterization of these experiments to improve the performance results.

Keywords: Grid, Bioinformatics, Performance analysis.

1 Background and problem statement

Over the past few decades, the combined improvement of the molecular research and information technologies has produced a remarkable amount of data related to molecular biology. In the beginning of the 'genomic revolution', the bioinformatics field pursued the creation and maintenance of databases to store biological information, such as nucleotide and amino acid sequences. The efforts were focused not only on design issues but also in the development of complex interfaces whereby researchers could both access existing and new data. However, the concept of bioinformatics currently aims to analyze and extract valuable information from this biological data through the creation and improvement of databases, algorithms, computational and statistical techniques and theory, to solve formal and practical problems.

One major research area in the bioinformatics field is the sequence annotation for studying the functionality of the different genes and regions. It involves computing the comparison of genomic and proteomic sequences with respect to annotated databases.

One of the most popular tools for performing the alignment of novel sequences is BLAST (Basic Local Alignment Search Tool) [1] from the NCBI (National Centre for Biotechnology Information) [2] which finds regions of local similarity between sequences. Based in a heuristic approach, this tool is able to obtain results several times faster than other approaches, such as dynamic programming or the Smith-Waterman algorithm [3]. BLAST emphasizes speed over sensitivity and thus, it cannot guarantee the optimal alignments. However, this emphasis on speed is vital to make the algorithm practical on the daily increasing databases. In fact, from its creation, GenBank [4] has doubled its size about every 18 months (similar to the Moore's Law for hardware), increasing the computer power needed by software applications. Although this tool could be inefficient for many cases and there are other improved tools in the literature (based on Hash Tables, based on the Burroughs-Wheeler transform, and a combination of both), the reliability of its results has given it a good reputation among Bioinformatics users.

The NCBI provides publicly a web-service implementation of BLAST to perform the homology search of few sequences against the ones contained in their databases (with a limitation on the number of simultaneously queries). This process, even using large databases, takes only few minutes. However, a normal use case requires performing the comparison of tens of thousands of sequences and it constitutes a very computationally intensive process. For instance, determining the importance of horizontal gene transfer in the evolution of eukaryotes requires performing the comparison of all prokaryote proteins with all the plant, animal and fungus proteins independently [5]. The computer (and sometimes storage) resources needed to tackle with this kind of experiments exceed the capabilities of conventional research groups (tens of CPU computation years following a sequential approach).

Nevertheless this kind of experiments can be massively parallelized and executed by means of e-Science infrastructures, such as EGEE [6], EELA [7] and some NGIs, following 'High-Throughput' approaches. In order to intensively reduce the makespan of the experiment, this paradigm proposes to split the work into thousands of loosely-coupled tasks, which are executed concurrently in the nodes of the Grid Computing infrastructure.

Unfortunately, since Grids are composed of heterogeneous resources, the lack of reliability is a major concern. On one hand, some jobs will simply fail due to different reasons, such as: bad configurations, software incompatibilities and resources availability. On the other hand, some other jobs could be hung up appearing to be running in the resources. So, in order to improve the efficiency of this kind of experiments, it is mandatory to detect these unproductive jobs and cancel them, instead of waiting for their finalization. Upon cancellation, the jobs must be immediately resubmitted to the most productive resources. However, differencing between long-running jobs and hung up jobs poses a notable challenge. As a consequence, the necessity of making estimations about the jobs' elapsed time arises.

In order to make predictions about the jobs' elapsed time it is important to take into account, among others, the following aspects: the computing resource features, the

current system load and the computational cost of the tasks. Our prediction approach exposed in previous works [8] is based on homogenizing the computational cost of the tasks combined with a static scheduling. In this way, the same wall clock or maximum life time is assigned to all jobs, regardless of the resources where they will be executed or the dynamic load of the infrastructure. Thanks to this measure, the response time and the failure rates are improved (in comparison with other simplistic approaches) but still the speed of the experiment is heavily penalized due to the resubmission of the last pending jobs. In fact, as it is showed in [9], the experiment completion achieves the 90% at the 50% of the global response time.

So, the motivation behind this work is to improve these results by formulating a model which approximates accurately the execution of BLAST jobs in a Grid environment.

The remainder of this paper is structured as follows. The second section starts exposing the contribution to the state-of-the-art. Then, the third section enumerates a set of parameters, likely to be relevant for predicting the turnaround time of a job. Next, the fourth section describes some parameter-sweep experiments carried out in a real Grid production infrastructure, to check the importance of the previous parameters. After that, the fifth section offers the procedure to obtain the performance model. Finally, the sixth section outlines the most important conclusions derived from this work.

2 State-of-the-art

Most of the works about improving the performance of BLAST focus on the implementation of different parallel versions. When parallelizing BLAST, there exist two main methodologies: the 'High-Performance' paradigm and the 'High-Throughput' paradigm. The High-Performance paradigm is the schema used by tools such as mpiBLAST [9] and TurboBLAST [10], which fragments the database into several fragments and distributes them to each processing node. This way, each node only needs to search a given query against its assigned portion of the database. On the other side, in the High-Throughput schema (the one most suitable for the Grid) the database to search is common to all nodes and the input sequences file must be partitioned and distributed. In this case, each node will be responsible for only comparing the sequences of its assigned input file fragment.

These techniques allow speeding up the execution of BLAST, but the heterogeneity present in the Grid infrastructures becomes an obstacle in realizing user desired levels of service. In order to deal with this issue, some works [11] analyze, in detail, resource and application dependencies for BLAST; while others [12] focus in the performance characterization at the micro-architecture level.

The work presented in this paper contributes not only with the identification of significant parameters in a BLAST execution, but also with the definition of a specific performance model. For instance, Quality of Service frameworks [13] will benefit from the increase in predictability offered by this model.

3 Relevant factors

The performance of BLAST sequence alignments is influenced by several factors that can be broken into three groups: factors related with the features of the computing infrastructure, factors related with the characteristics of the input data and the execution parameters. Next, all these factors are commented exhaustively.

3.1 Grid computing infrastructure

Grid infrastructures offer access to a large number of coarsely-linked and heterogeneous resources, coming from multiple administrative domains. These infrastructures normally are composed of a group of key resources whose operation logically will have effect on the jobs' performance, and can be summarized as:

- CE (Computing Elements): The access point to a farm of identical computing nodes, which contains the LRMS (Local Resource Management System). The LRMS is responsible for scheduling the jobs submitted to the CE, allocating the execution of a job in one (sequential) or more (parallel) computing nodes. In the case that no free computing nodes are available, jobs are queued. Thus, the load of a CE must be considered when estimating the turnaround of a job.
- WN (Working Nodes): Each one of the computing resources accessible through a CE. The expected performance of a job being executed in a particular CE will vary in function of the characteristics of its WNs. The factors of a Working Node, identified as determinant, are: SPECint, SPECfp and the memory size.
- SE (Storage Element): Resources in which a task can store long-living data to be used by the computers of the Grid. In the scenario presented in this work, the database to search by all jobs will be replicated in a carefully chosen group of SEs. This practice is necessary due to the size limitation imposed by current Grid middlewares in the job file attachment (10 Megabytes in the gLite [14] case). The most critical [15] and time-consuming preprocessing task of any BLAST job consists on retrieving the target database from a SE. Of course, the bandwidth between the CE hosting the job and the SE will play an important role in the final response time.
- LFC (Logic File Catalog): A hierarchical directory of logical names referencing a set of physical files and replicas stored in the SEs.
- BDII (Berkley Database Information System): Service point for the Information System which registers, through LDAP, the status of the Grid. Useful information relative to CEs, WNs and SEs can be obtained by querying this element.

These previous terms will be referenced along the text.

3.2 Input data

As it was mentioned before, a massive BLAST experiment involves performing the alignment or homology search of a huge number of uncharacterized sequences with respect to the ones contained in an annotated database. However, following the High-Throughput paradigm, the duty of a single job consists on searching only the sequences in its assigned input-sequence chunk (resulting from the fragmentation pre-processing task). Thus, according to this reasoning, the computational cost of the jobs will be expressed as a function of the input file sizes (the input-sequence segment and the database).

Additionally, the behavior of the BLAST algorithm introduces another possible relevant factor, related to the input data. As it was pointed out in the introduction, BLAST approximates the Smith-Waterman algorithm by using a heuristic approach, instead of performing the full alignment (which is a lot more time-consuming). In this manner, the comparison of two tentatively not similar sequences can be skipped. Therefore, the similarity or discrepancy degree between the input-file sequences and the target sequences poses a new factor that must be considered in the performance analysis.

3.3 Execution parameters

The performance of any software is affected by the input arguments, normally introduced by the user via the command line interface. The BLAST algorithm is no exception and it includes a parameter which could have great influence on its execution time. This parameter is known as blast hits or simply ‘bhits’ and it indicates, for each sequence in the query file, the maximum number of results which can be placed in the output file.

4 Experimentation

This section describes, extensively, the experiments performed in order to verify the significance of the factors outlined in the previous section. Subsequently, those parameters identified as relevant will be employed to formulate accurately the performance model.

4.1 Infrastructure used

EGEE (Enabling Grids in E-sciencE) constitutes the largest production Grid infrastructure in the world, using as middleware gLite 3.2. A Virtual Organization (part of EGEE) oriented to biomedical applications, ‘biomed’, has been selected taking into account its consolidation, and the wide range of available computing resources (more than 139336 cores simultaneously accessible).

4.2 Data used

The data used for testing purposes are protein sequences coming from UniProt [16], the most comprehensive protein sequence public database, and used in previous works [5]. UniProt is offered divided into 9 groups: archaea, bacteria, fungi, human, invertebrates, mammals, plants, rodents and vertebrates. Based on the file size of these databases, the archaea (56 MB) and invertebrates (562 MB) groups were considered adequate for acting as input-sequence file and database, respectively. It is important to remark that these groups were selected with the intention of having lower and higher cases.

4.3 Experiment 1: Varying the input data size

The experiment performed for checking the influence of the input data size entailed the creation of several input sequence files and databases with different sizes but similar average sequence length. In concrete, for this test were produced: 10 input sequence files ranging from 0.5 to 9 MB (with an invariable step of 0.5 MB) and 9 databases from 50 to 450 MB (with a constant step of 50 MB), involving the execution of 90 tasks in total. Evidently, all these jobs were executed in the working nodes of the same CE, which have the same performance features.

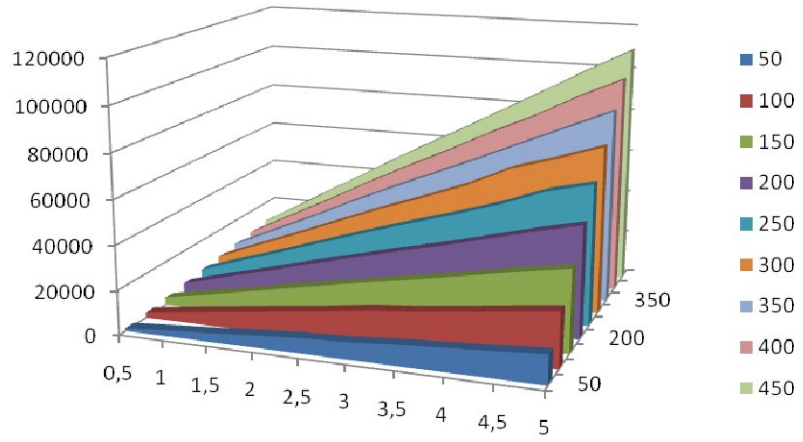

Fig. 1. Response time (seconds in the vertical axis) of BLAST tasks executed in the same computing node, using different sizes of the input file (MB in the bottom axis) and the database (MB in the vertical right axis).

At first sight, the results (depicted in Figure 1) show a nearly linear relation between the input-sequences file size and the database file size. Indeed, the more input and target sequences to process, the more processing time is required.

4.4 Experiment 2: Varying the degree of resemblance

The second experiment pursues to appreciate how the turnaround of a job is affected by the degree of resemblance between the input and target sequences, as it was pointed out in Section 3.2. For that purpose, the first step involved producing three new versions of one of the databases generated in the previous point. The procedure for generating these databases consisted on simply replacing 1%, 5% and 10% of their contents respectively, with coherent random values (letters corresponding to the four nucleic acid bases that make up DNA).

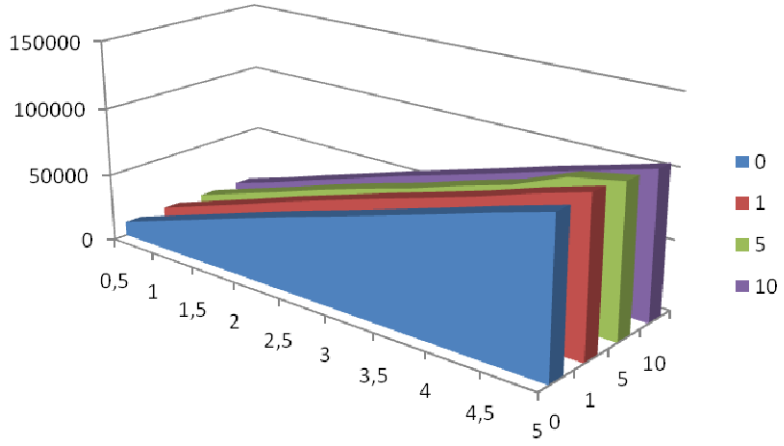

Fig. 2. Response time (seconds in the vertical axis) of BLAST tasks executed in the same computing node, using different sizes of the input-sequences file (MB in the horizontal axis) and the amount of content altered in the database (% in the right axis).

As it is shown in Figure 2, the results are practical identical which means that the time consumed by a job is independent from degree of similarity between input and target sequences. Thus, this factor will not be taken into account in the final performance model.

4.5 Experiment 3: Varying the parameter 'bhits'

The third experiment was intended for checking how the execution time of a BLAST job is affected by the value of the argument 'blast hits'. For that purpose, the analysis involved performing a small sweep-parameter experiment. All the jobs in this experiment were executed using the same input files (the input-sequences file of 5MB and the database of 450 MB) but different value of 'bhits'. More precisely, the minimum value was fixed to 5000 hits and the maximum to 25000 hits, with a step value of 2500. In addition, with the objective of minimizing the effects of anomalous

results, each task was executed several times and the average execution time was computed for each of them.

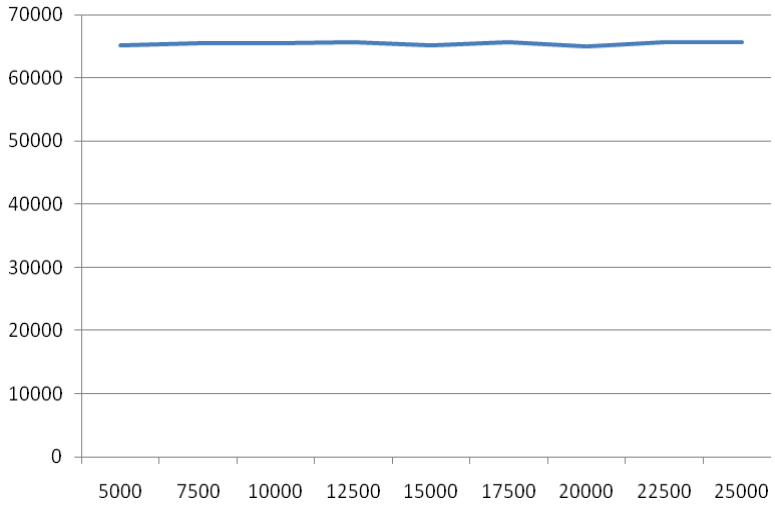

Fig. 3. Execution time (seconds in the vertical axis) of BLAST tasks executed in the same computing node, using different values of 'bhits' (horizontal axis).

As it can be appreciated in Figure 3, the value of the response time is approximately 65000 for all the submitted jobs. Thus, it can be concluded that the value of the parameter 'blast hits' has no influence in the computational cost of a BLAST job. So, as in the degree of similarity case, this factor will not be taken into consideration when defining the final performance model.

4.6 Experiment 4: Varying the computing node

The aim of the last experiment is to analyze the influence of the factors related to the WN's performance. With this goal in mind, the performance values of the CEs of 'biomed' VO have been retrieved by means of the BDII. The operation of the BDII, as any Grid Information Service, is governed by the standard GLUE Schema [17]. This schema defines which data must be provided by the information resources and it has been adopted by different Grid projects, such as EGEE or NorduGrid. In particular, the BDIIs employed in EGEE offer three helpful values for this study: the benchmarks SpecInt2000 and SpectFloat2000 and the size of the memory cache. Using this information, 19 CEs with different speeds and memory sizes were chosen for hosting the testing jobs.

Upon completion of all the jobs submitted in this experiment, the first step was to study the correlation between the SpecInt and SpecFloat benchmarks and the performance results, lying in the standard outputs of the jobs.

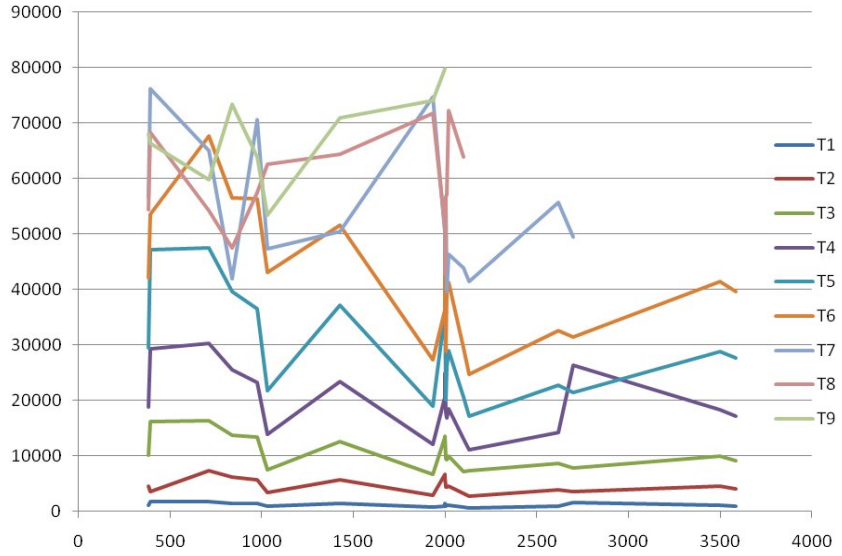

Fig. 4. Response time (seconds in the vertical axis) for 9 different combinations of the input file size and the database file size in function of the average Spec2000 (units in the horizontal axis).

In order to observe the existent correlation between the values of the Spec2000 benchmarks and the actual performance of BLAST, a graph like the shown in Figure 4 has been designed. This Figure shows the response time (retrieved from the outputs of the jobs) for 9 different types of jobs (combinations of the input file size and the database size) in function of the Spec2000. The Spec2000 value is computed as the arithmetic mean of SpecInt2000 and SpecFloat2000.

At first glance it can be stated that the SpecInt and SpecFloat benchmarks are not correlated with the actual performance for BLAST: the higher the value of the Spec2000 of the node doesn't necessarily mean the lower the response time observed. Actually, there are several situations in which a priori better performance results on a worst response time.

To reinforce the previous proposition, the Pearson product-moment correlation coefficient has been calculated for each type of job (see below).

Table 1. Correlation indexes for the 9 types of jobs considered.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
Pearson	-0,433	-0,309	-0,533	-0,42	-0,521	-0,549	-0,413	0,079	0,55
p	0,052	0,175	0,014	0,06	0,017	0,011	0,081	0,772	0,089

In view of the results offered by Table1, it can be asserted that there is a low linear dependency between the Spec2000 benchmarks and the BLAST performance for any type of job: Pearson's r ranges from 0.08 and 0.55 (very distant from the extreme values 1 and -1). In addition, the significance is very poor for all the jobs. It is especially noticeable the case in which the relation between both variables is due to nearly pure randomness (77% of probability).

Behind this inexistence of correlation could there be various reasons.

The first one refers to the unsuitability of the benchmark for BLAST. The BLAST program is a very file-intensive process, i.e. a noteworthy portion of the execution time is spent on I/O operations (writing the alignment results in the output file). So, a conventional benchmark, such as Spec2000, could be an inaccurate reference for estimating the turnaround of a BLAST task.

Another important factor for explaining the lack of correlation observed in Figure 4 is the fact that Spec2000 is a commercial benchmark. As a consequence, instead of actually running the benchmark, the administrators of many sites choose to merely obtain the values from the tables corresponding to the available processor. In the worst case, the values of the benchmarks are simply not published. Certainly, all these measures negatively affect a performance study in the Grid.

So, at this point it is necessary to obtain a new performance estimator. Instead of relying on the values of the benchmarks, this new estimator will be based only on the computing times observed for each node. Thus, it seems reasonable to call this new parameter: the empirical estimator. In concrete, the empirical estimator proposed in this work is the ratio of average performance with respect to a fixed node. In order to compute the values of the new estimator, it is necessary to obtain, for each node, the sum of all the computing times (considering the same number and class of jobs). Finally, choosing a concrete node as a reference, all the speed-ups/downs are computed.

Figure 5 graphically shows the correlation between the response time and the new estimator. However, analogous to the benchmark case, Table 2 quantifies the existent correlation between the empirical estimator and the computing time, via the Pearson's r.

Table 2. Correlation indexes for the 9 types of jobs considered, with the empirical estimator.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
Pearson	0,88	0,788	0,957	0,897	0,978	0,96	0,889	0,691	0,73
p	1,86 E-07	3,33E-05	6,65E-11	8,72E-08	2,10E-13	3,12E-11	8,33E-07	0,004	0,015

In contrast to the results offered by Table 1, Table 2 shows a more linear dependency between both variables (the computing time and the performance estimator proposed). Obviously, the significance is also higher, with values always above 98%, but generally above 99,99%.

With respect to the memory size, the dependence observed was very low, thus excluding this parameter will be excluded from the final model.

In conclusion, these results corroborate one of the hypotheses proposed at the start of this sub-section: the performance of the node affects the response time needed by a

BLAST job. Logically, this factor will be pertinently included in the final performance model.

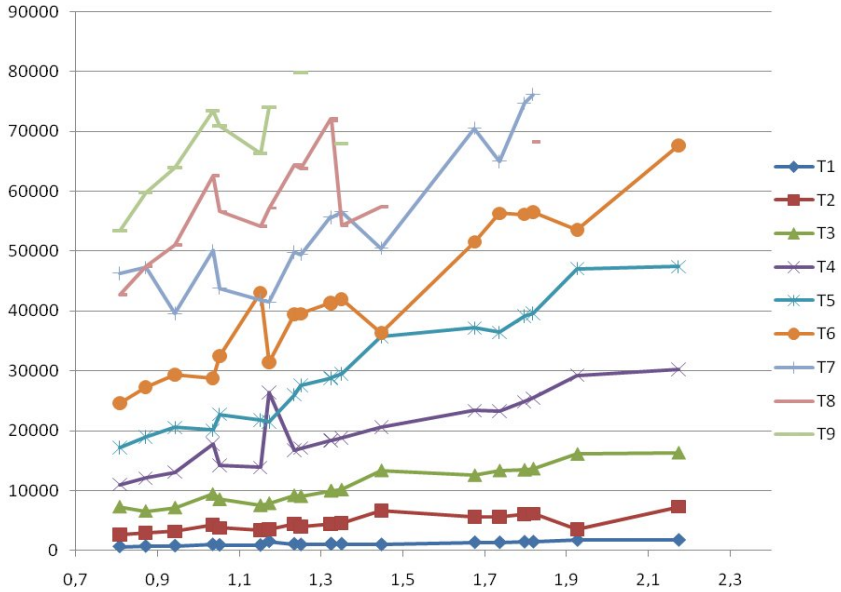

Fig. 5. Response time (seconds in the vertical axis) for 9 different types of jobs in function of the ratio of average performance with respect to a fixed node (units in the horizontal axis).

5 Performance model

Before proposing a model, it will be very helpful to recap all the conclusions extracted so far.

First, in Section 4.2, it was concluded that there is a linear dependence on both types of input data: the input-sequences file and the database. Then, Section 4.3 demonstrated how the degree of resemblance between the sequences doesn't affect the computing required to process them. Additionally, in Section 4.5 it was showed that the parameter 'bhits' also doesn't influence the response time of the tasks. Finally, in the Section 4.6, it was proved that there is a direct dependence on the processor speed factor.

Accordingly to the exposed above, the proposed model is shown in Formula 1.

$$T = \frac{P_{Sp}}{Sp} P_{Tm} \left(\frac{T_{inp}}{T_{inp\_basal}} \frac{T_{BD}}{T_{BD\_basal}}\right) + K \,. \qquad \textbf{(1)}$$

Where:

- P_{Sp}, P_{Tm} and K are the unknown parameters for the regression model.
- T_{inp} and T_{BD} are the fixed values for the size of data and database.
- Sp is the ratio between the response time in a reference site and each other site.
- T_{inp_basal} is 0,5 and T_{BD_basal} is 50.

The procedure for estimating the unknown parameters included in Formula 1 was divided in two phases.
The goal of the first step was to determine the parameters only related to the data size, i.e. P_{Tm} and K. Hence, focusing on the available data for the reference site, Formula 2 was used as a model for linear regression.

$$T = P_{Tm}\left(\frac{T_{inp}}{T_{inp\_basal}}\frac{T_{BD}}{T_{BD\_basal}}\right) + K . \quad (2)$$

Through a typical estimation method, such as OLS (Ordinary Least Squares), the resulting values for P_{Tm} and K were 835,62 and 1065,5 respectively.
Next, it was necessary to complete the prediction model by estimating the parameter related to the performance of a site: P_{Sp}. With all the data from the 19 sites, it was defined an overdetermined system: a great number of equations with only one unknown (P_{Sp}). Making use of the least squares method, the optimal solution for the parameter P_{Sp} was found (Formula 3).

$$T = \frac{1.35}{Sp}835.62\left(\frac{T_{inp}}{T_{inp\_basal}}\frac{T_{BD}}{T_{BD\_basal}}\right) + 1065.5 . \quad (3)$$

To appreciate the accuracy of the performance model, Figure 6 and 7 show a comparison between the real and predicted execution time.

6 Conclusions

Estimating the response time is a key issue for achieving an efficient load balancing and reducing the failure rate of massive BLAST experiments. This work describes a set of experiments with the intention of defining a performance model that can be used to accurately estimate the response time of such tasks.

Studying the data retrieved in the experiments, it was found that there is a direct dependence with the data size, the reference database and the performance indicator. However, due to the inaccuracy inherent to the benchmarks published by the sites, the

performance indicator had to be redefined. In its place, it was used an indicator based in the observed values: the empirical estimator. Other factors, such as the memory size, the degree of resemblance and the number of hits were found not relevant for this study.

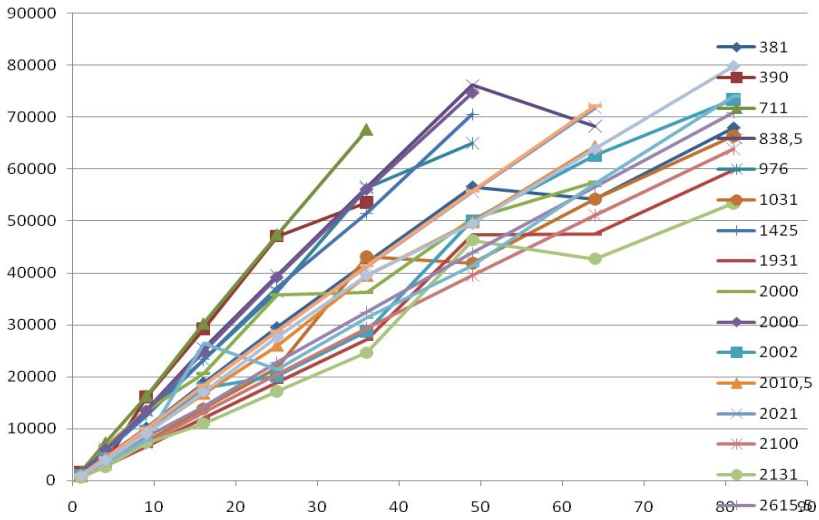

Fig. 6. Real execution time (seconds in the vertical axis) for the 19 sites considered in function of the type of job (size of the input data with respect the basal size).

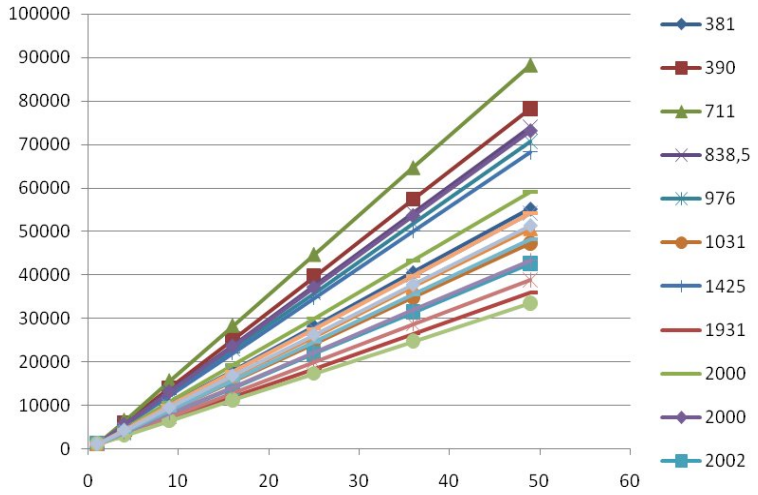

Fig. 7. Estimated execution time (seconds in the vertical axis) for the 19 sites considered in function of the type of job (size of the input data with respect the basal size).

Upon finishing the identification of the significant factors, the linear regression and the least square methods were used for estimating the parameters of the performance model. Although the applicability of this model might be restricted to the infrastructure, target of this work, the methodology for obtaining a prediction model is still valid for any other environment.

In the future, this work will be completed introducing parameters from other components, such as the Workload Manager, the LRMS and the bandwidth between CEs and SEs.

References

1. Altschul, S.F., Gish, W., Miller, W., Meyers, E.W., Lipman, D.J.: Basic Local Alignment Search Tool. J. Mol. Biol. 215, 403--410 (1990).
2. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov
3. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195--197 (1981).
4. Benson, Dennis A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A, Wheeler, D.L. : GenBank. Nucleic Acids Research, vol. 28, No1, pp. 15--18 (2000).
5. Blanquer, I., Carrión, A.A., Hernández, V., Forment, J., Conejero, V.: Estimating the horizontal gene transfer between prokaryotes and plants by using e-Science infrastructures, pp. 49-58, CIEMAT (2009).
6. EGEE: Enabling Grids for E-SciencE, http://www.eu-egee
7. EELA: E-Infrastructure shared between Europe and Latin America, http://www.eu-eela.org
8. Aparicio, G., Blanquer, I., Hernández, V.: A Highly Optimized Grid Deployment: the Metagenomic Analysis Example. In: HealthGrid'08, Studies in Health Technologies and Informatics, vol. 138, pp 105-115, IOS Press (2008).
9. MpiBLAST, http://mpiblast.lanl.gov
10. Bjomson, R.D., Sherman, A.H., Weston, S.B., Willard, N., Wing, J.: TurboBLAST: A Parallel Implementation of BLAST Built on the Turbo Hub. In: Parallel and Distributed Processing Symposium, International, vol. 2, pp 0183, Boston (2002).
11. Afgan, E., Bangalore, P.: Performace Characterization of BLAST for the Grid. In: IEEE 7th International Symposium on Bioinformatics & Bioengineering, pp. 1394-1383 (2007).
12. Sanchez, F., Salamí, E., Ramirez, A., Valero, M.: Performance analysis of sequence alignment applications. In: Proceedings of the IEEE International Symposium on Workload Characterization, pp. 51-60 (2006).
13. Blanquer, I., Hernández, V., Segrelles, D., Torres, E.: A supporting infrastructure for Evaluating QoS-specific Activities in SOA-based Grids. In: International Conference on Distributed and Parallel Systems, pp. 167-178, Springer (2008).
14. gLite: LightWeight Middleware for Grid Computing, http://glite.web.cern.ch/glite
15. Blanquer, I., Carrión, A.A, Hernández, V., Pignatelli, M., Tamames, J.: Improving the execution of Bioinformatics applications by using pilot jobs. In: 3rd Iberian Grid Infrastructure Conference Proceedings, pp. 43-53, Netbiblo SA. (2009).
16. UniProt: The Universal Protein Resource, http://www.uniprot.org
17. GlueSchema, http://glueschema.forge.cnaf.infn.it/