

# Metrics for Grid Applicability: a Distributed Elliptic Curve Platform Assessment

Paulo Trezentos<sup>1\*</sup> and Arlindo L. Oliveira<sup>2</sup>

<sup>1</sup> UNIDE / ISCTE  
Edif. ISCTE, Av. Forças Armadas  
1600-082 Lisboa [Paulo.Trezentos@iscte.pt](mailto:Paulo.Trezentos@iscte.pt)

<sup>2</sup> INESC-ID / IST  
R. Alves Redol 9  
1000 Lisboa [aml@inesc-id.pt](mailto:aml@inesc-id.pt)

**Abstract.** The distribution of computational load among several nodes is an important step for problems requiring High Performance Computing (HPC). This phase is critical because bad decisions could cost time and money.

The emergence of heterogeneous networks with resources available for a wide group of users as provided by Grid Computing [3] creates the need for formal processes and metrics to evaluate if the application of the problem to the Grid is feasible.

In this article we introduce some auxiliary indicators for measuring the potential applicability of a parallel application to a Grid. Using the measures defined in the internal draft (GWD-I) produced by the Network Measurement WG [6] from the Global Grid Forum and RFC 2330 [7], the authors present some aggregate metrics that are useful in the characterization of the parallel applications that are well adapted to existing grids.

The defined auxiliary metrics were useful in the validation of a concrete application that factorizes numbers using the elliptic curves method (ECM) [5] over a testbed Grid. The results of the application of the metrics to this specific mathematical algorithm are presented.

## 1 Introduction

Not all parallel implementations are adequate to all grids. We usually face one of these two scenarios: having a parallel application developed and build a grid for running it or having an already existent grid and implementing a parallel solution suitable to that grid.

In the first case, the application is “fixed” and the grid should serve its purposes. This is generally the case of large problems where the necessary funds are put at the team’s disposal.

---

\* The author would like to thank ADETTI and CFC by their support to this work, and the means that were put at his disposal.

The second case applies to already existent grids where you can not change its network and hardware capabilities. We then have a “fixed” grid and our application should be developed specifically having in mind the grid specifications.

This work proposes the use of aggregated measurements that can help the algorithm designer in the process of determining if a given grid is suitable for a given problem. The auxiliary ratios hereby presented should not be used in a deterministic way as a “yes or no” oracle suggestion. They are useful to perform grid / application specification analysis and avoid a situation where the definition of requirements is based only in “common sense experience”.

The use of combined metrics is not new (See, for example, [2]). Nevertheless, the authors believe that the requirements and specification analysis stage still lacks the help of appropriate tools.

## 2 Auxiliary Metrics

The auxiliary metrics that will be presented in this section measure the applicability of one distributed application to a Grid. We assume that we have a typical grid and application scenarios:

- **Grid** - a set of computing resources (both hardware and software) combined in units called nodes that are interconnected through a network infrastructure; if a client / server architecture is enforced by the application, some of the nodes can act as clients (processing back-ends) and some as servers (front-ends);
- **Application** - program that runs simultaneously in different nodes of the grid with a specific goal to be achieved in limited time. Some interaction between nodes is supposed to exist. In the rest of this document the analysis will assume that the application is composed of a server program that delivers data to be processed and by the clients that, after processing the data, reply to the server with the results and, optionally, send a request for more data.

### 2.1 Critical Factors

The first step consists in defining which critical factors should weight in an equation that ponders if the grid is suitable. Due to space requirements, we will not address in detail the process by which these variables are measured.

- **minimum bandwidth required** - what is the minimum bandwidth between nodes that the application requires;
- **maximum latency allowed** - what is the maximum latency / delay between nodes that the application supports;
- **data transmission frequency** - how often there is a need to transmit data between nodes;
- **amount of data to transmit** - which amount of data is transmitted in each communication.

Having these variables in mind, some combined measurements were produced.

## 2.2 Frequency of Interruptions for Communication (FIC)

This auxiliary metric analyzes the average number of suspensions in application processing due to the need to establish a communication to exchange data.

We will use *minute* as unit but that should be flexible enough to allow to use subunits when more appropriate.

$$\frac{\text{Node processing capacity (FLOPS)} * 60}{\text{Total application floating point instructions}} \quad (1)$$

By **node processing capacity** (FLOPS) we represent the number of floating point instructions that one node of the grid is capable of processing by second, on the average.

**Total application floating point instructions** means the average number of floating point instructions that a client node application processes between receiving the data for processing and delivering the results to the server. Since the number of floating point instructions might depend of the data received, an average value should be used.

The result of the equation will be the number of *interruptions per minute*.

Moreover, this auxiliary measurement reflects the granularity of the compromise between transmitting more data in each transfer and having a longer processing period or the reverse.

Note that this granularity can, for some type of applications, be flexible but, for the rest, is fixed, and one can not increase or decrease the amount of data to be transmitted.

## 2.3 Total Time of Data Transmission (TTDT)

The **Total Time of Data Transmission (TTDT)** is the absolute time period that it takes the server to transmit data to the client.

This value is given by the following equation:

$$\frac{\text{Total data to be transmitted (KB)} * 8}{\text{Bandwidth of the connection (Kbps)}} \quad (2)$$

**Total data to be transmitted** (KB) quantifies - in KBytes - the quantity of data that will be transferred from the server to the client for processing purposes.

By **Bandwidth of the connection** (Kbps) is meant the bandwidth availability as defined in NMWG GWD-I [6]. The result will be in seconds, and multiplication by 8 is required so that both measures work with the same unit (KB).

We assume that:

- in what concerns this ratio, latency is negligible compared to the total time of the transmission; for latency control purposes we will introduce later another ratio (MCO);

- communication is only established between server and client, not taking into account the client-to-client connections and the client-to-server results transmission; this last factor can commonly be neglected since the transmission of results is usually not bandwidth intensive;
- the initial transfer of the client program into the grid node is not pondered since it happens only once.

If any of the previous points is not applicable, then the equation should take into account the changed conditions.

## 2.4 Maximum Communication Overhead (MCO)

**Maximum Communication Overhead** is a limit that should be established indicating the maximum overhead that we are willing to accept for the communication.

For instance, if we draw the limit of MCO to 5% this will mean that we are ready to accept that our system will spend at most 5 minutes in communication for each 100 minutes of processing. We can check if an implementation does not cross this limit by using the following equation:

$$= \frac{\text{TTDT (seconds)}}{\frac{1}{\text{FIC}} * 60} * 100 = \frac{\text{TTDT (seconds)} * \text{FIC}}{60} * 100 \quad (3)$$

## 2.5 Maximum Latency Overhead (MLO)

The **Maximum Latency Overhead** represents the ratio between the delay introduced by the network constraints and the absolute time of the overall transmission<sup>3</sup>. This indicator can help us establish a limit (%) for the maximum latency overhead that we are ready to accept.

The MLO can be defined as follow:

$$= \frac{\text{Latency/delay}}{\text{Overall time of data transfer}} * 100 \quad (4)$$

The latency / delay can be defined [6] in two ways: one-way delay and round-trip delay. Moreover, since it is defined by network and protocol characteristics, we can use the RFC 1242 [1] definition. In this case, it is defined as the average time that one bit takes to be transferred from the original computer's port into the port of the destination computer.

## 2.6 “Goodness frontier” for the Application

Using the previous combined metrics we can think in a Cartesian axis that uses two of the indicators as axis and Maximum Communication Overhead (MCO) as the limit to the use of the application over the grid.

<sup>3</sup> This time is different from the previously introduced TTDT since it already includes latency and other delays reflecting the overall time of the communication.

The line that represents the frontier can be defined using the following coefficients:

- **A** - maximum limit to TTDT above which it does not pay to use the grid, even for small values of FIC. To define its value we can take into account the latency through the MLO ratio;
- **M** - slope of the *goodness* straight line

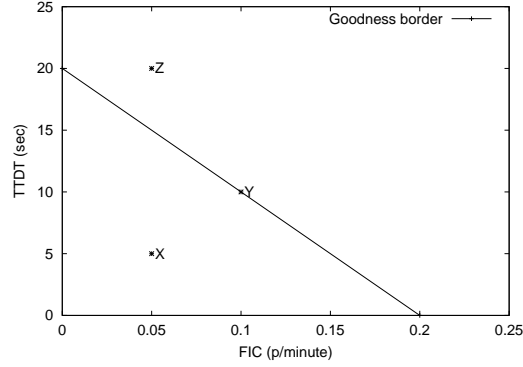
The slope can be found through:

$$m = \frac{A}{FIC} - \frac{TTDT}{FIC} \quad (5)$$

Thus:

$$TTDT = A - M * FIC \quad (6)$$

The straight line will then reflect our defined MCO. Figure 1 exemplifies the



**Fig. 1.** “Goodness” application ratio

use of this ratio. The value of FIC is represented in the **X-axis**. In this axis, points closer to the origin represent less interruptions, and, therefore, a better chance of applicability to the grid.

In the **Y-axis** we represent the Total Time of Data Transmission (TTDT). In the same way, points closer to the origin are points better adapted to use the grid.

The *goodness frontier* is drawn using  $A = 20$  and  $M = 100$ . This frontier (represented as a dotted straight line) is specially interesting when we have to take a decision about employing or not the grid in our application, because it considers simultaneously variables that characterize both grid and the application.

We can illustrate the use of these measures with some examples:

1. The X point, located at (0.05,5) is below the goodness frontier. As such, we can say that it represents an application that is clearly appropriate to the grid. Its location (0.05,5) comes from an FIC of 0.05, reflecting an interruption in each 20 minutes, and an TTDT of 5 seconds, which corresponds to a good working point.
2. The Y point (0.1,10) is near the threshold of the *goodness frontier*. With an FIC of 0.1 - one interruption in each 10 minutes - and an 10 seconds TTDT, the decision about its feasibility can not be clearly taken.
3. The Z point (0.05,20) has the same FIC as the X point but a much higher TTDT. This fact puts it above the goodness frontier and therefore in the non-applicability zone.

We emphasize that the cited combined metrics take into account several factors, among which the processors performance. One grid with exactly the same specifications of another grid but with slower CPU nodes will have a better result in what concerns the applicability of a given application since it will take more time to process the same set of instructions. Thus, before exercising these indicators we should verify if the CPU computational power is suitable to our problem.

### 3 ECM over the Grid (DISTECM)

An Elliptic Curve over a field K is the set of solutions that respect:

$$Y^2 = X^3 + AX^2 + BX + C \quad (7)$$

The points in the curve form an Abelian Group on which all operations are well defined. The negative of any point  $P_1 = (x_1, y_1)$  is defined by  $-P_1 = (x_1, -y_1)$ . The Elliptic Curve Method (ECM) was applied to factorization problem proposed by Hendrik Lenstra [5].

ECM has become more popular than Pollards P-1 since it works well even if  $P - 1$  is not smooth. Assume that  $n > 1$ ,  $\gcd(n,6)=1$  and that  $n$  is not a prime power with power  $> 1$ . The ECM algorithm for factoring  $n$  consists in [5]:

1. Draw  $a, x, y \in Z/nZ$
2. Identify  $P=(x:y:1) \in N$  and select an integer  $K=K(a,b)$
3. Calculate  $K \cdot P$
4. If it fails, the  $d$  divisor of  $n$  it was found. If not, we return to point 1 again and start over.

The platform devised for Distributed ECM uses the Globus Middleware<sup>4</sup>. Globus provided grid services such as Information (GRIS / GIIS), Security (GSI), Resource Management (GRAM) and Data Management (GASS). These services can be used to launch processes, retrieve output files and allow single sign-on on the grid.

<sup>4</sup> <http://www.globus.org>

At application level, we implemented a client-server program where the communication is performed using standard sockets and not PVM or MPI (MPICH). The server role is to distribute curves among the several clients until one of them finds factors of  $n$ .

The platform was tested for the factorization of the Partition Number P(10341). The idea of using partition numbers for factorization was launched by the RSA company<sup>5</sup>. The number P(10341) has 109 digits and was rapidly factored using the distributed platform.

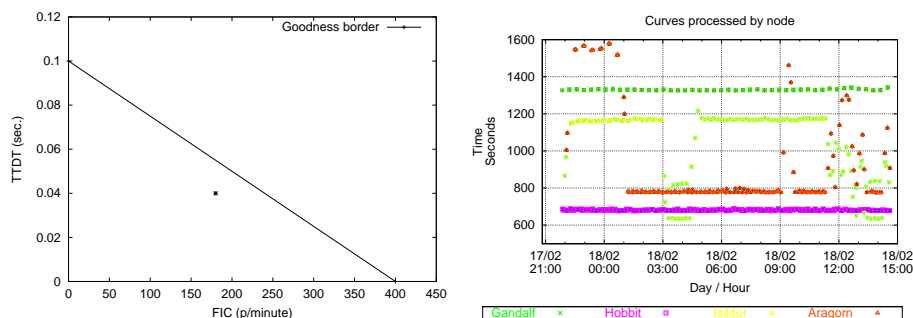
The factorization took 17 hours and 7 minutes with 8 clients over a small SMP cluster with 4 nodes. Table 1 shows some statistics of the load distribution observed.

Node	# curves	Aver. Time	Std. Dev.
Gandalf	91	1330 "	3,93
Hobbit	177	682 "	2,95
Aragorn	133	904 "	245,57
Isildur	117	1028 "	194,2

**Table 1.** Summary of curves processed by each server

## 4 DISTECM Assessment

In this scenario our grid is based on a Beowulf Cluster [8]. A Beowulf is a cluster of PCs built with commodity components and off-the-shelf hardware solutions [4]. PCs are usually connected to a 100Mbps FastEthernet network or Gigabit. This corresponds to a 100Mbps connection, with an average latency estimated at 6ms. Experimental evaluation of the (average) CPU speed gives a value of 52.4 MFLOPS. The “goodness frontier” for this grid environment is represented in the left of figure 2. In this graphic, the “Goodness frontier” limits FIC to 400



**Fig. 2.** Result of “Goodness frontier” metric application

<sup>5</sup> <http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html>

interruptions per minute which is acceptable for a FastEthernet network. As the marked point is below the “goodness frontier” we conclude that the DISTECM would be suitable for this kind of grid. The experimental results have indeed shown an average CPU occupation of 97.7%. This indicates a well balanced load distribution, and reasonably small idle times, which validates experimentally the applicability of the metrics proposed.

The rightmost graphic in figure 2 illustrates the distribution of tasks over SMP nodes. Together with the high occupation ratio observed, it shows that, despite the differences in CPU speed, the available processing speed is used up to its full capacity.

## 5 Conclusions

Mathematical algorithms like ECM can be suitable or not for distributed processing over a grid. The metrics hereby presented can be of help in the process of defining a formal approach to understand what applications are appropriate for use in grids.

The distributed ECM (DISTECM) platform has proved to be a good approach for factoring numbers under 100 digits. An NFS distributed platform based on CWI code that broke an RSA 512 bits key was not presented in this article but was also developed by the authors and is more efficient in the factorization of numbers above 100 digits.

Future work will concentrate in enhancements of these metrics in the Global grid Forum (GGF) scope and application of the same to other fields of problems.

## References

1. S. Bradner. RFC 1242: Benchmarking terminology for network interconnection devices, July 1991. Status: INFORMATIONAL.
2. T. Ferrari and F. Giacomini. Network monitoring for grid performance optimization. *Computer Communications*, submitted for publication, 2002.
3. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.
4. P. T. José Guimarães. Spino: A distributed architecture for massive text storage. *ICEIS*, 1:244–248, 2001.
5. H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.
6. B. Lowekamp, B. Tierney, L. Cotrell, R. Hughes-Jones, T. Kielmann, and M. Swany. A hierarchy of network measurements for grid applications and services - draft. Technical report, Global Grid Forum - Network Measurements Working Group, July 2002.
7. V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330: Framework for IP performance metrics, May 1998. Status: INFORMATIONAL.
8. J. Radajewski and D. Eadline. Beowulf howto. Technical report, Linux Documentation Project, Nov. 1998.