

Grid Result Checking

Cécile Germain-Renaud
Laboratoire de Recherche en Informatique
cecile.germain@lri.fr

Dephine Monnier-Ragaine
Laboratoire de l'Accélérateur Linéaire
monnier@lal.in2p3.fr

ABSTRACT

Result checking is the theory and practice of proving that the result of an execution of a program on an input is correct. Result checking has most often been envisioned in the framework of program testing or property testing, where the issue is the conformity of the program to some a-priori specification. Very large scale distributed computing systems demand to tackle the issue of computation correctness, albeit from hypothesis very different from the program testing ones. The general issues examined in this paper are the following. First, the definition of checking methods adapted to large-scale Monte-Carlo simulations; for these applications, no external criterion can be used to assess the quality of the result. Second, two result checking algorithms which minimize the overall overhead through an adaptive strategy. Finally, a specialization of this framework to a case study, the Auger astrophysics experiment. Our main contributions are: first to focus on checking Monte-Carlo simulations, which have rarely been considered previously; second to define a probabilistic checking strategy including the risk of first kind (false positive) as well as the risk of second kind (false negative) which is usually the only one considered, and which is compatible with Byzantine saboteurs; third, to exploit the probable characteristics of the behaviour of the saboteurs to optimise for the most frequent case. Finally, we show on a case study that the implementation details can be carried out.

Categories and Subject Descriptors

D.2.4 [Software Engineering]: Software/Program Verification; J.2 [Physical Sciences and Engineering]: Physics

General Terms

Verification Experimentation

Keywords

Grids, Simulation, Result Checking

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CF'05, May 4–6, 2005, Ischia, Italy.

Copyright 2005 ACM 1-59593-018-3/05/0005 ...\$5.00.

1. INTRODUCTION

Result checking [3] is the theory and practice of proving that the result of an execution of a program on a given input is actually correct. Result checking has most often been envisioned in the framework of *program testing* [2] or *property testing* [13], where the issue is the conformity of the program to some a-priori specification. Very large scale distributed computing systems demand to tackle the issue of computation correctness, albeit from hypotheses very different from the program testing ones.

Grids can be broadly classified in two categories: Institutional Grids (IG), and Global Computing or P2P (GCP) systems. IG extend the computing centre model to a large geographical scale, based on the virtual organization paradigm [10, 9]. GCP [11, 8] harvest the computing power provided by individual computers, the *collaborators*, to run very large and distributed applications.

GCP such as SETI@home [18] or XtremWeb [7] use uncontrolled computing resources, and are thus very vulnerable to *sabotage*: some of the workers may tamper with the computation process and report false results. Although few GCP were actually deployed in an open environment, at least three of them (distributed.Net, SETI@home, Decryphon) suffered from this kind of attack. The case of SETI@home is well documented; some collaborators substituted for the original FFT a faster but inadequate one; the probable goal was to get a better classification in the SETI's "hall of fame".

IG have better control over their collaborators; with respect to result checking, the main difference is that the collaborators are identifiable through strong authentication systems. Nevertheless, IG target broad scientific collaborations where very various individuals, groups and institutions, carry out many digital simulations and where the chain of the data processing is only partially automated. An example of this kind of grids is EGEE/LCG, one of the largest grids deployed so far. Formerly, the simulations were run in a very small number of computing centres, ensuring relatively easy traceability of software. A more decentralized and autonomous access to computing resources is now considered important even in particle physics [5]. The possibility of errors in the process is then increased.

We will use the following terminology (as defined by [19]): *falsification* stands for the process which creates erroneous results, and *saboteur* for either a machine or a human being tampering with the computation. In increasing order of malicious intention, the origin of falsification may be running an incorrect code, or a correct one with incorrect inputs,

whether inadvertently or to get speedup; or the will to skew the results; both GCP and IG are exposed to these risks. Only GCP might suffer from *free riding*, where the collaborators use the resources, but do not provide any; GCP might also be exposed to *denial of service* attacks, where the goal is to halt the GCP system by destroying its credibility.

Two classes of strategies can be used to prohibit or limit falsification: a-priori prevention or a posteriori checking. Prevention enforces the use of the proper software and inputs. For IG, prevention relies on organization management and is out of the scope of this paper; however, as pointed before, the need for autonomous use of the grid limits this approach. For GCP, at user level, the issue is very similar to the protection of execution against malicious agents, for which code encryption and execution tracing has been proposed [14]; however, both imply large overheads for designing and running the encrypted code. At the system level, the code and data fidelity can be enforced by a chain of certification, which ultimately relies on hardware (TCPA-Palladium [15], microcode [17]), but with major sociological problems.

Besides, none of these methods is relevant for open source code. Thus only a posteriori checking will be considered here.

The general organisation this paper are as follows. Section 2 sketches the general issues raised by checking large-scale Monte-Carlo simulations; for these applications, no external criterion can be used to assess the quality of the result. Section 3 proposes two result checking algorithms, which minimize the overall checking overhead through an adaptive strategy, together with a quantitative evaluation of their performance. Section 4 proposes a framework for exploiting these algorithms when both identified and anonymous collaborators contribute to the system. Section 5 describes the specialization of the oracle component of the algorithms to a case study, the Auger astrophysics experiment.

Our main contributions are:

- to focus on checking Monte-Carlo simulations, which have rarely been considered previously [22];
- to define a probabilistic checking strategy including the risk of first kind (false positive) as well as the risk of second kind (false negative) which is usually the only one considered ; this strategy is compatible with Byzantine saboteurs;
- to exploit the probable characteristics of the behaviour of the saboteurs to optimise for the most frequent case;
- to show on a case study that the implementation details can be carried out.

This paper extends the ideas presented initially in [12] on three points. The first one is to present a comprehensive cost model; the second one is to show that Byzantine saboteurs can be dealt with; the last one is to propose a case study.

2. OVERVIEW

2.1 Grid Applications

In this paper, we consider only *multi-parameter* or *pleasantly parallel* applications, which consist of a large set of independent jobs. Most result-checking or property-testing

work is rooted in exhibiting a characteristic property of the computation; this property is able to separate the correct results from the incorrect ones. For many important grid applications, like Monte-Carlo simulations or the search for rare events, this model does not apply to jobs.

- Monte-Carlo methods build a sample of a statistical law. In this respect, they differ from numerical methods which *e.g.* compute the solution of a partial differential equation (PDE) which fall into the category of one-way invertible functions. In many Monte-Carlo simulations used in physics and biology, the program simulates a complex system, whose only local interaction laws are known. There almost never exists a positive property allowing validating an individual job, and re-execution is required for job testing. On the other hand, the results are exploited by statistical procedures, in order to estimate the parameters or the form of the distribution of quantities of interest. Such procedures are amenable to fault tolerance: *robust* statistical methods tolerate aberrant values of unknown amplitude, if their number is known. The checking algorithm has thus to guarantee an upper bound on the falsification rate.
- The search for rare events is exemplified by SETI@home, or the search for Mersenne prime numbers. The frequency of positive results ("extraterrestrial signal found") is supposed to be extremely low. The issue is to check the mass of negative results. Such applications share with the Monte-Carlo ones the characteristic of requiring re-execution to prove correctness. However, they differ in the fact that the individual results, not the collective behaviour, are the point of interest ; thus these applications are essentially non fault-tolerant.

For both cases, the only possible verification of individual jobs is re-execution. Whatever scheme is applied (vote, *m*-first voters), the penalty of an exhaustive checking is at least doubling the execution cost.

Fault-tolerance is exhibited by other applications, for instance ray-tracing. For non fault-tolerant applications, test strategies will have to target the elimination of unreliable collaborators, thus are restricted to IG, or to accept the penalty of multiple re-execution (this strategy has been applied by SETI). The rest of this paper considers only the fault-tolerant applications.

One advantage of this quality control approach is its genericity: as the semantics of the jobs does not matter, it will be implemented as the level of grid middleware, and be independent of the user algorithms.

2.2 A Cost Model

The previous discussion leads to define a verifiable unit, a *batch*, as a group of jobs. A checking algorithm accepts or rejects a batch as a whole; thus all the jobs of a batch have to wait for the completion of the checking algorithm before any of them can be validated.

A batch of size n can be modelled as a vector of random binary variables (x_1, \dots, x_n) . $x_i = 0$ with probability p (resp. 1 with probability $1-p$) if job i is correct (resp. false). Assuming independent jobs (x_1, \dots, x_n) follows a binomial law:

$$P_p(x_1, \dots, x_n) = p^{S_n} (1-p)^{n-S_n}, \text{ where } S_n = \sum_1^n x_i.$$

Independent jobs do not imply independence of saboteurs: the checking algorithms will sample the batch to assess p ; if the sample is uniform over the batch, the model of independent jobs is always applicable (provided that the size of the sample is small with respect to the batch one). A counterexample would be a test which would choose the first n jobs issued by the system as its sample, when the saboteurs are faster than the honest collaborators. Therefore the independent jobs model can accommodate Byzantine adversaries by enforcing uniform sampling.

The application fault tolerance is defined by two parameters: p_a is the tolerated rate of falsification in a batch, and ϵ is the tolerated risk of false negative: ϵ is the risk which the user agrees to take by using a system which guarantees its results only by a probabilistic algorithm. The checking algorithm \mathcal{A} decides if the falsification rate p is greater than p_a , with a risk ϵ of optimistic error. Here and afterwards an *error* is a wrong decision of \mathcal{A} , either rejecting a good batch or accepting a bad one. \mathcal{A} has access to an oracle, which deterministically asserts if a particular job has been falsified or no. The typical oracle is re-execution on dependable machines, or vote.

When \mathcal{A} must reject a batch, all the executed jobs are useless, since a batch is delivered entirely or not at all; thus the cost of \mathcal{A} must take into account, not only the cost of the tests, but also the overall resource usage. To model the resource usage, let B be the size of the batch, $s(p)$ the number of calls to the oracle and $r(p)$ the number of jobs executed before a rejection. For a given value of the falsification rate p , the overhead of \mathcal{A} is then $s(p)$, whatever final issue (acceptance or rejection), plus $r(p)$ when \mathcal{A} rejects, normalized by the batch size:

$$C(p) = \frac{1}{B} (s(p) + r(p)P_p(\text{reject})). \quad (1)$$

However, p is by hypothesis unknown. Moreover, it is not true in general that the overhead increases with p , as shown by the adaptive algorithms presented in the next section. Thus the comparison or the parameterisation of the checking algorithms must take into account the statistical distribution of p . Let Q be this distribution. Then a performance indicator for \mathcal{A} is the average overhead \mathcal{C}_a under Q when \mathcal{A} should accept :

$$\mathcal{C}_a = \frac{1}{B} E_Q[C(p)|p \leq p_a] = \frac{1}{BQ([0, p_a])} \int_0^{p_a} C(p)dQ, \quad (2)$$

For instance, a saboteur-free system is by definition a system where p is always null; thus $Q(\{p = 0\}) = 1$. On the other hand, any reasonable test must accept systematically perfect batches, *i.e.* $P_p(\text{reject}) = 0$; therefore $\mathcal{C}_a = s(0)$: the overhead is precisely the number of calls to the oracle.

\mathcal{C}_a measures the overhead in a *stable* state. A stable state is a state where the system can deliver, *i.e.* $p < p_a$. When the system should not issue anything, because the falsification rate is higher than acceptable, the only sensible cost is the complexity of the halting procedure, measured by the number of calls to the oracle. Detection of this situation must lead to corrective actions, which are sketched in section 4.2.

The distribution Q models the behaviour of saboteurs.

- Q concentrated on 0 and 1: $Q(0) = q, Q(1) = 1 - q$, describes the alternation of massive and coordinated

attacks (all jobs falsified, p is always 1) and the ideal situation (p is always 0).

- Q as a power law: $Q_\lambda(p \leq x) = x^\lambda; 0 < \lambda \leq 1; x \in [0, 1]$, represents a distribution which concentration on the null rate decreases with λ , down to the uniform case ($\lambda = 1$) where all falsification rates are equally likely.

3. ADAPTIVE ALGORITHMS

Three distinct behaviours can be considered.

- *Normal.* Most collaborators are not saboteurs. This is true if the system can control the origin of jobs, or if sabotage is made difficult (binary encrypted code).
- *Massive attack.* The saboteurs systematically falsify their results.
- *Subtle attack.* In this scenario, a large number of collaborators are saboteurs. If these collaborators were to falsify systematically (massive attack), the system should be able to eliminate them and continue working with the remaining honest collaborators. Therefore the most subtle attack consists in providing collaborators who falsify only a fraction of results, and who are thus difficult to identify.

The possibility of very different scenarios is an argument for *adaptive* checking algorithms, which optimise for the most frequent situation. The realization of a subtle attack is complex: the entity responsible for sabotage must control a significant fraction of collaborators, organize anonymity, and falsify not systematically. The checking algorithm must therefore be optimised to normal and massive attacks scenarios, while being able to detect a situation of subtle attack, at higher cost.

3.1 Wald's sequential test

The fault-tolerance property of Monte-Carlo simulation leads to the broad class of statistical decision procedure known as *hypothesis testing*. What is needed is to assess if the falsification rate is below some given threshold p_a .

Parametric hypothesis testing is a procedure which decides, given a distribution P_p (here the binomial law) with p unknown, if $p \leq p_0$ or $p > p_1$, p_0 and p_1 being known values. The specification of a test is its *acceptance region* H_0 (null hypothesis); for an empirical sample (x_1, \dots, x_n) of P_p , the test accepts if and only if $(x_1, \dots, x_n) \in H_0$.

Two risks must be considered: α is the probability (under P_p) to reject when p is actually less than p_0 (false positive); β is the probability (under P_p) to accept when p is actually greater than p_1 (false negative).

H_0 is constrained by the fact that for $p \leq p_0$, the test must accept with probability greater than $1 - \alpha$; for $p \geq p_1$, the test must reject with probability greater than β ; in the region $p_0 < p < p_1$, the quality of the outcome of the test is not quantified; thus $[p_1 - p_0]$ measures the zone where the test is meaningless. n is the *sample size*.

A standard result of statistics (Neyman-Pearson theorem) shows that there is an optimal test among the non-adaptive tests, with samples of *fixed* size. The principle is to define the acceptance region by the number of defects in the sample:

$$H_0 = \{(x_1, \dots, x_n) | S_n \leq c\}.$$

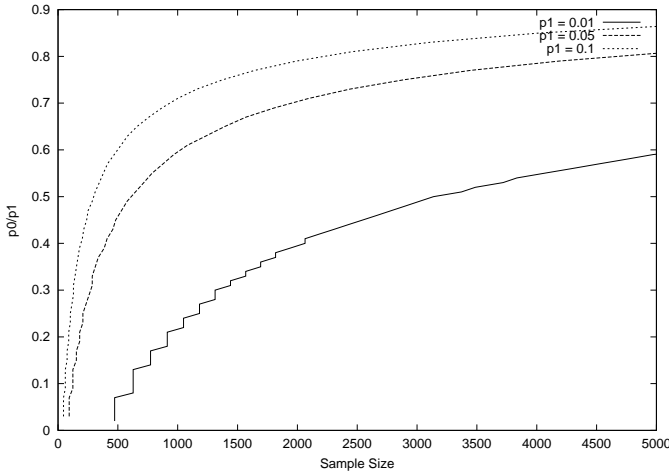


Figure 1: Non adaptive testing - binomial law

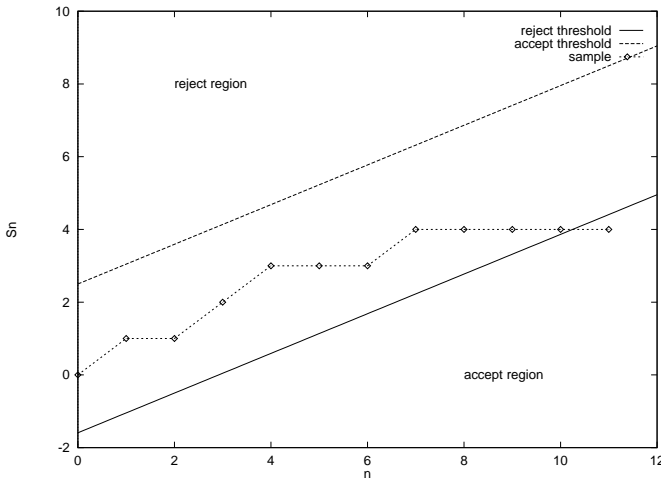


Figure 2: Principle of the sequential test.

c and n , which define the test, are determined by $P_{p_0}(H_0) \geq 1 - \alpha$ and $P_{p_1}(H_0) < \beta$.

This test is offered in all the statistical environments, from spreadsheets to high performance statistical systems. Its cost is very high, as shown in Fig. 1: to get a relatively low accuracy ($p_0 = p_1/2$), the size of the sample is in the order of 100 for $p_1 = 0.1$, but of several thousands for $p_1 = 0.01$. The sequential test defined by Wald [21, 23] formalizes the intuitive idea that a partial test is sufficient if the previous data imply acceptance or rejection. Fig. 2 describe the algorithm: the test begins with a sample of size 1; at step m , if the number of defects in the sample (x_1, \dots, x_m) is less (resp greater) than a value which depends of m , the test accepts (resp. rejects); if the number of defects is between these two values, the sample size is increased. The sample size becomes a random variable, and the test cost is measured by $E_p(n)$, the average sample size under P_p . It can be shown that the test ends ($E_p(n) < \infty$) [23]. The choice of the sequential test has two advantages over the static test (Fig. 3):

- On average, its cost is less than that one of the static test, at all defect rates.

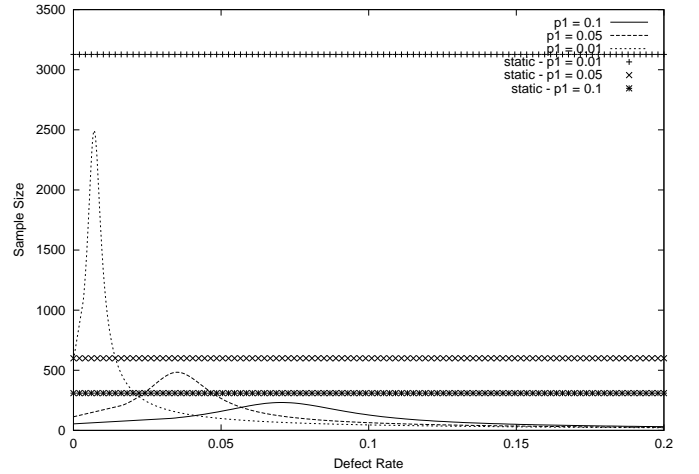


Figure 3: The sequential test. Sample size for the binomial law. For all examples, $p_0 = p_1/2$.

- It is adaptive; the cost increases to a maximum between p_0 and p_1 , and drops fast after. The benefit in comparison with the static test is a factor of 5 for the normal case, where there are very few saboteurs (p close to 0), and of several orders of magnitude for a massive attack (p close to 1) .

In the following, $L(p)$ is the probability that Wald's test accepts. By construction, $L(0) = 1$, $L(p_0) = 1 - \alpha$, $L(p_1) = \beta$ et $L(1) = 0$. Other properties of Wald's test are recalled in the appendix.

Sequential tests are routinely used in various areas. Therapeutic trials must end as soon as significant data is available, for ethical reasons (when one treatment is significantly better than other one), which motivates a considerable literature concerning variants of the test (for a bibliography see [6]). In industrial process control, the non-adaptive test corresponds to V-tests, and the sequential test to CUSUM methods. [21] notes the relation between the sequential test and dynamic programming. Finally, the notion of *credibility* developed by [19] revolves around the same ideas, but with elementary mathematical tools.

3.2 Checking a Batch

The most difficult case is the one where the collaborators are anonymous: the composition of a batch cannot be improved by the removal of suspicious collaborators. This section will present two algorithms adapted to this case.

The Simple Sequential Test

The first algorithm is a direct application of Wald's sequential test. For a falsification rate p , the sampling cost $s(p)$ is $E_p(n)$, and the probability of rejection is simply $1 - L(p)$. To comply with the user specifications, it is necessary to choose $p_1 = p_a$ and $\beta = \epsilon$, ensuring that the upper bounds on the falsification rate and false negative will not be exceeded.

In this framework, the number of calls to the oracle $s(p)$ is the size of the sample, that is the random variable n ; the number of jobs executed before a rejection is simply the batch size B . Thus, according to (2),

$$C_a = \frac{E_Q[E_p(n)|p \leq p_a]}{B} + E_Q[1 - L(p)|p \leq p_a].$$

The complete specification of algorithm requires to define p_0 , α and the size of the batch B . The batch size is mainly constrained by external factors, such as the maximal time a user would accept to wait for the results; the four parameters α , β , p_0 and p_1 are redundant to define the actual behaviour of a sequential test (which is fully defined by three parameters, *e.g.* the common slopes of the acceptance and rejection lines, and their intersection with the Y-axis). Thus in the following, we focus on the dependance on the p_0 parameter.

In the previous expression, the $E_p(n)$ term increases with p_0 , for p fixed, and the second term decreases: a test is more selective, in acceptance, if p_0 and p_1 are closer, but requires a larger sample. These inequalities remain true with Q -averages.

Fig. 4 display the overhead computed by numerical integration, as a function of p_0 , and for the power law probability distributions on p presented in 2.2. For this example, $\alpha = \beta = 5\%$ and $B = 1000$. The leftmost column corresponds to $\lambda = 0.01$ (falsification distribution concentrated to 0); the middle column is $\lambda = 0.1$; the rightmost column is $\lambda = 1$ (uniform distribution). The lower row corresponds to $p_a = 0.1$, the upper row to $p_a = 0.01$. The horizontal axis is p_0 .

The most obvious result is that this simple test provides acceptable performance in the normal case, when the distribution of p knowing that $p < p_1$ is highly peaked at 0. On the other hand, the simple test becomes very costly in situation of subtle attack, modelled by an uniform distribution: sabotage consists then in providing enough wrong results to cause high resource use, and enough good results to prevent the system from efficient reaction. The simple test is therefore very vulnerable to a denial of service attack.

With the cost model described before, the overhead has two components: the number of oracle calls on one hand, and of erroneous rejections on the other hand. Fig. 4 displays the breakdown of these two components (dark grey = erroneous rejections, light grey = number of calls). The simple test remains static, in the sense that it does not really integrate the possibility of early rejection. On the other hand, a strictly sequential implementation of Wald's test would not be convenient for servicing multiple users. The next section proposes a tradeoff between the fully parallel test (simple test) and a sequential implementation.

A two-phases test

An early detection of a rejection situation decreases the overhead, since the batch is not entirely executed. A 2-phases test starts by assessing the reaction of the system toward a particular batch, by testing all the results of an initial subset. When a decision is reached, if positive, the rest of the jobs is launched, then a new simple test is performed by random selection amongst the results of the whole batch. The batch is accepted only if both tests accept.

If the first step rejects, the number of calls to the oracle is $s_1(p) = E_p(n)$; if it accepts, the number of calls to the oracle is $s_2(p) = 2E_p(n)$. Thus, $s(p) = (1 + L(p))E_p(n)$. Rejection at the first step costs only the execution of the fraction of the batch required for reaching a decision, thus $r_1(p) = E_p(n)$; if the first step accepts and the second one rejects, all the batch is executed, thus $r_2(p) = B$. Hence, $r(p) = (1 - L(p))E_p(n) + L(p)(1 - L(p))B$. Finally, we get

$$C^{2\text{phases}}(p) = \frac{2E_p(n)}{B} + L(p)(1 - L(p)).$$

The second test is necessary, the first jobs not being representative of the system, except if the adversaries are independent. A much weaker hypothesis than the one of independent adversaries is to assume that the falsification rate remains constant between the first and the second phase, what is equivalent to assume that the adversary cannot discover the frequency of tests. Under this hypothesis, the values of β for the 2-phases test must be adapted, with $\beta = \sqrt{\epsilon}$, to get the same overall risk of the second kind.

When there are not falsifications ($p = 0$), with the adapted β , equation (3) in the appendix gives $E_p^{2\text{phases}}(n) = E_p(n)/2$; as $L(0) = 1$, $C^{2\text{phases}}(0) = C(0)$, therefore the 2-phases test overhead is not worse than the simple one.

As soon as the falsification rate is significant, Fig. 5 and 6 show that the 2-phases test significantly decreases the overhead for a large scale of p_0 values. In these examples, the upper (resp lower) graph corresponds to $p_a = 0.01$ (resp. $p_a = 0.1$). $\alpha = \beta = 5\%$; medium and high are as in Fig. 4. At low falsification rate, the results for the overhead are very close, thus the figures were not included. The two-phases test has therefore two advantages.

- The system is much more robust with respect to subtle attacks.
- The range of tradeoffs between the cost of rejection and the oracle calls is much larger.

If the hypothesis of a constant falsification rate is not valid, therefore assuming an omniscient adversary, the situation is more complex [12]. The simple test is better when the falsification rate is concentrated at 0; for $p = 0$ the overhead of the two phases test is double of the one of the simple test; if the falsification rate is uniform, the two phases test is better for small values of p_0 , as in the previous case.

4. THE CREDIBILITY OF THE COLLABORATORS

In a simpler case, the association between jobs and collaborators can be tracked down. It is then possible to test not only the batches, but also the collaborators themselves, and to eliminate jobs coming from suspicious collaborators. This rejection must not be immediate, because an unintentional falsification is likely to be transitory (erroneous manipulation).

4.1 Simultaneous checking

For simultaneous testing of collaborators and batches, the adequate setting is to accept (with probability $1 - \alpha$) any correct production, therefore to set $p_0 = p_a$. The definition of the other parameters (β and p_1) will be provided by external constraints, for instance the ratio between the test size and the subset of the batch (\mathcal{B}_r) delegated to collaborator r (the overall batch \mathcal{B} is the union of all \mathcal{B}_r). The size of \mathcal{B}_r being limited, the simple sequential test is probably the only applicable one. The test outcome is double: first, the admittance of rejection of \mathcal{B}_r ; second, an assessment of the collaborator credibility; this assessment is binary (dependable or suspicious). Both results can be effectively exploited.

Concerning the assessment of \mathcal{B} , the impact of the test is to improve the quality of the batch results: any results coming from suspicious collaborators are discarded, and their jobs are rescheduled to collaborators that have passed the

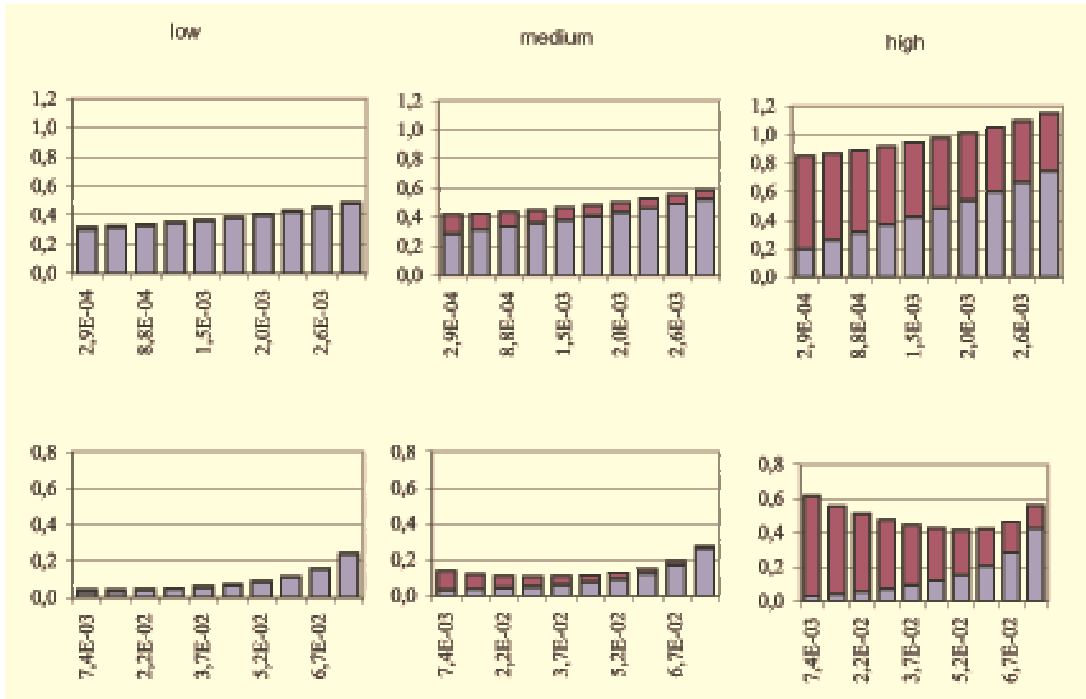


Figure 4: Overhead for the simple test

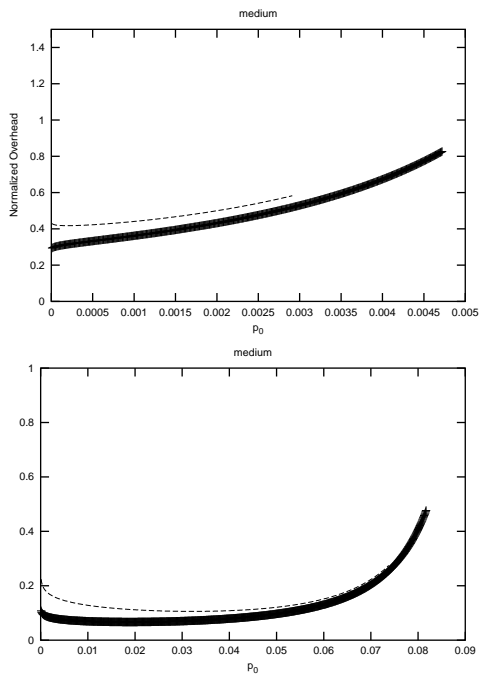


Figure 5: Comparison of the overheads for the simple test (dotted) and the two phases (bold) test - Medium falsification rate.

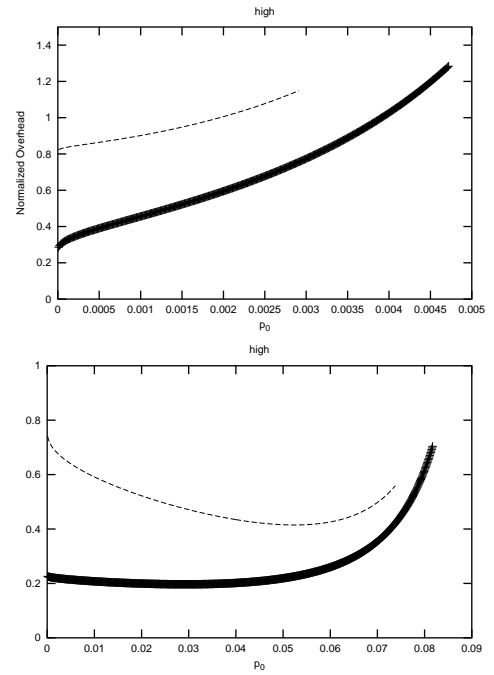


Figure 6: Comparison of the overheads for the simple test (dotted) and the two phases (bold) test - High falsification rate.

previous test. When \mathcal{B} is complete, the results come only from the remaining non suspicious collaborators. The batch must then be re-tested with $p_1 = p_a$, to guarantee conformity with the prescribed tolerance; as the batch is very likely to be of very good quality, a very low rejection threshold p_0 is reasonable, allowing for a fast test.

4.2 A Reliability Architecture

The checking algorithms must be inserted into a more general process, the *inspector*. The general objective of the inspector is to dynamically allocate resources as a function of, on one hand the user quality requirements (accepted falsification rate and error tolerance) and on the other hand the behaviour of the system. A realistic modelling of the inspector would require more information on attack scenarios than what is currently available; we will only provide some initial ideas.

The resources can be categorized threefold: the absolutely reliable collaborators, which are under the direct control of the system administration, the identified collaborators, and the others ones. The primary role of the absolutely reliable collaborators is checking individual jobs. As their number is necessarily small, they constitute the inspector bottleneck. Extending the pool of collaborators entitled to check to identified collaborators, is a by-product of the test sketched in 4.1: the reliability of the collaborators itself is a random binary variable, thus subject to the hypothesis testing. However, an erroneous promotion of a computing collaborator to checking collaborator would have very serious negative impact on the inspector. Thus the most adequate statistical test for the promotion would probably be a test of rare events rather than a sequential test. Nevertheless, the promoted collaborator must continue being tested by random re-execution of its results on fully reliable collaborators.

A much more difficult problem is recovery from attacks. When the falsification rate is estimated higher than the acceptable threshold, continuing operations without particular intervention would simply mean trying to realize the statistically unlikely event of false negative. For IG systems, alerting the collaborators is probably enough. For GCP systems, one simple but useful tool is to enforce reloading the application code (eg the FFT in SETI) from a reliable source. For the identified collaborators, only saboteurs will have to reload. For the anonymous collaborators, reloading might lead to an indirect denial of service, the reliable sources becoming overloaded by simultaneous requests. Thus the reloading has to be scheduled over time, from the assumed identities of the collaborators.

5. A CASE STUDY

5.1 The Pierre Auger Observatory

The Monte-Carlo simulations in the Auger experiment are an interesting and probably general example of the possible improvements to the general scheme exposed in the previous sections. The Pierre Auger Observatory studies the very high-energy cosmic rays. These cosmic rays are electromagnetic particles with energies above 10^{19} eV (by comparison, accelerators attain at best 10^{12} eV). They are also extremely rare: above 10^{19} eV, the arrival rate is only 1 event per km^2 per year; the especially interesting ones with energies above 10^{20} eV have an estimated arrival rate of just 1 per km^2 per century. The origin of these cosmic rays is a major scientific

issue: in the standard model of the universe, such events should be so rare (GZK cutoff) that the accidental detection of a few of them in the 90's was rather unlikely. The main goal of the observatory is to record a significant sample (statistics in physicist language) of high-energy events.

Very energetic cosmic rays are observed in the earth atmosphere. The primary energetic cosmic ray collides with a nucleus in the air, creating many secondary particles, which share the original energy. The secondary particles also collide with nuclei in the air, creating a new generation of still more particles that continue the process. This cascade, called an *extensive air shower*, arrives at ground level with billions of energetic particles that can be detected over approximately 10 square kilometres. Auger particle detectors are deployed with a separation of 1.5 km on a 3000km^2 surface in Argentina, in order to sample each air shower's density at numerous locations on the ground. Fluorescence detectors register the shower development in the atmosphere.

The observation of cosmic rays is thus very indirect. Physically relevant information, particularly the nature of the primary electromagnetic particle, its initial energy and its angle, must be inferred from the measurements of the physical effects produced in detectors by the shower. The reconstruction models are designed from *in silico* experiments: numerical simulations of showers and detectors. Shower simulations dominate the computation time for these simulations, and will only be considered in the following.

5.2 Air shower simulation

The simulations inputs are the physical parameters (primary particle, energy, angle), plus steering parameters (eg thinning). The output is a sample of the shower development including ground level particles. The simulation codes are Aires [20] and Corsika [16].

The physics of a shower is a complex and partly random process, particularly because of the low density of the upper atmosphere. Consequently, the simulation is of the Monte Carlo type: the physics of a collision is deterministic, but the location of this collision is statistical. This first source of randomness comes from the modelling of physical reality. The constraints of numerical simulation add internal algorithmic randomisation. There are simply too many particles in shower (eg a 10^{20} eV primary particle creates a shower of 10^{11} particles). Statistical sampling of particles is governed by the *thinning* parameter. Table 1 shows a typical impact of the thinning parameter over the execution cost; in general, the computation time and storage requirements depend exponentially on the thinning. Decreasing the thinning decreases the statistical fluctuation due to internal randomisation.

5.3 Checking a shower

The previous sections detailed the testing methods, which provide an upper bound for the falsification rate inside a batch. What remains to be optimised is the cost of testing individual jobs, here shower simulations.

Principle

The goal is to design a shower checker better than re-execution. The exponential dependency on thinning allows to approximate a shower G (subject to check), by computing a control sample $\mathcal{S}(G)$ for the same physical input parameters, with a thinning large enough for an overall execution time

Relative Thinning	10^{-4}	10^{-5}	10^{-6}	10^{-7}
Computation time	1mn 43s	16mn 52s	2h 50mn	24h58mn
Storage	541KB	5MB	46MB	589MB

Table 1: Impact of thinning on simulation - Energy 10^{20} eV - Xeon 2,4GHz

of $\mathcal{S}(G)$ less than the execution time of G . The problem is then to decide if G and $\mathcal{S}(G)$ are compatible. This is an instance of the classical *outliers detection* problem in statistics. Numerous studies analysed the power of various tests against alternatives and tabulated the acceptance thresholds [1]. However, these tests can be rigorously derived only for parametric problems, where the distribution shape is known a-priori (gaussian, hyper-geometric etc.). For shower simulations, the double source of randomness leads to the combination of gaussian distributions due to internal randomisation and non-necessarily gaussian distributions coming from the above-mentioned physical effects.

Thus, the outlier detection test itself has to be designed from data mining of existing (and hopefully correct) shower results. This design is an ongoing project; only very preliminary results based on the physical interpretation of the result parameters and some crude statistical methods are exposed here.

Test design

The output of a simulation is a highly structured dataset. Therefore, direct tampering with the result files is difficult; moreover voluntary falsification is not a real threat in the Auger context. The normal mode of falsification would more likely be due to involuntary bugs in the submission process; it would substitute the results of a shower G' for those of another shower G . Thus, checking a shower amounts to prove that the *input parameters* of a given shower are what they were assumed to be. The next section will focus on checking the following input parameters: particle energy and particle type.

Post processing of a shower output delivers a set of physical quantities; these quantities do not capture all the useful physical information, but they depend in a complex manner on the whole dataset and therefore constitute good candidates for the test. The most significant quantities with respect to the *energy parameter* are probably those who represent the shower maximum: X_{\max} is the altitude of the point where the number of charged particles reaches his maximum and N_{\max} , the value of the maximum. The simulated shower initially expands, each collision producing new particles, then becomes diminishing, when the secondary particles have not enough energy to produce new particles; below a threshold called the cutoff energy, particles are not followed anymore. X_{\max} is not discriminant enough, because the impact of the altitude of the first collision introduces a large variability. On the other hand, the number of particle N_{\max} does not depend on the altitude of first collision.

However, experiments proved that the shower maximum is not discriminant for the *particle kind* parameter. A discriminant quantity must be searched amongst absolute values. For physical reasons, the number of muons at ground level N_{Muons} is the best candidate.

The test is the deviation from the mean: $\frac{|x-m|}{s} < c$, where m and s are respectively the mean and deviation estimators. The estimator of m is the median, and the estimator of s is

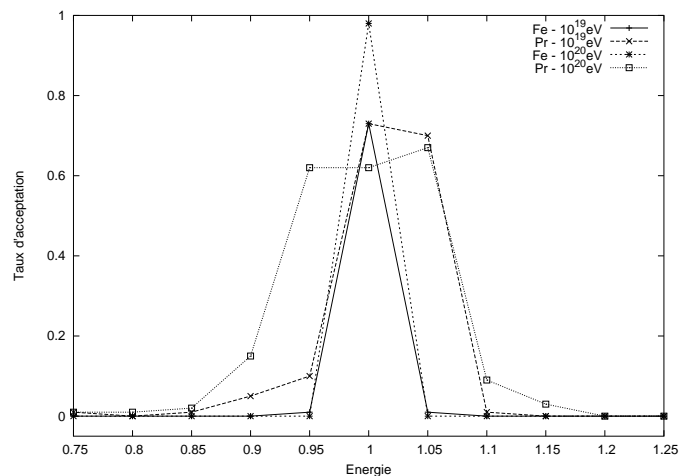


Figure 7: Test Performance - Energy

the unbiased one (for a gaussian distribution, this test is the maximum likelihood test). More precisely, the test accepts if $(\frac{|x_1 - m_1|}{s_1} < c_1) \wedge (\frac{|x_2 - m_2|}{s_2} < c_2)$, where the first test relates to N_{\max} and the second to N_{Muons} .

Experimental results

We used the Auger simulation database and the XtremWeb Global Computing system at LAL for shower simulations. The terms of experience are the following. The energy of showers to check varies from 0.75E19 to 1.25E19 (unit is eV) for the first example and from 0.75E20 to 1.25E20 for the second example; the kind can be either Iron or Proton. One shower G is thus, for instance, (Proton, 0.9E19). For each G , the control sample is composed of 20 showers, with the G parameters, except the thinning which is 10^{-4} .

The performance of the test is computed on a set of showers at 1.0E19 for the first example, and 1.0E20 for the second one. Each shower in this set play the role of a falsified shower. To have a significant sample of falsified showers, we had to use all the available showers in the central Auger database, with a 10^{-6} thinning. This is a worst case, because 10^{-6} is the lower bound of physically relevant simulations, and is due to the fact that the Auger experiment is still in an early phase. However, this provided sample sizes in the order of 100, except for (Iron, 10^{20}), where it is only 44.

Fig. 7 presents the acceptance ratio (number of shower accepted over total number of showers) for $c_1 = c_2 = 1$, when the only variable to be checked is energy. For instance, if G is (Iron, 0.85E19), the point on the curve Fe- 10^{19} eV with abscissa $X = 0.85$ gives the number of accepted showers from the 100 showers (Iron, 1.0E19) taken from the Auger database, divided by 100. The ideal result would be to have an acceptance ratio of 1 for showers with $X = 1$, and 0 for all the other ones. The test has a very good behaviour for

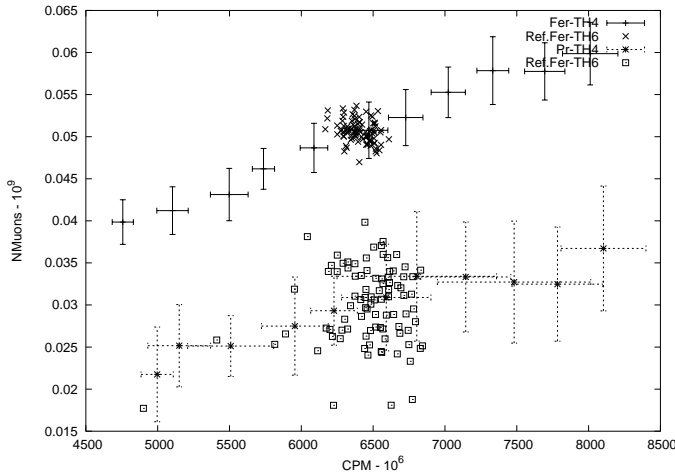


Figure 8: Distribution of Energy and Nmuons at 1.0E19eV

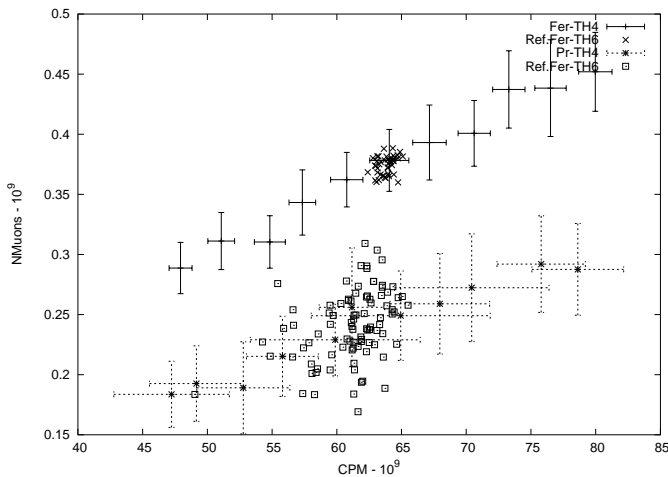


Figure 9: Distribution of Energy and Nmuons at 1.0E20eV

Iron showers, but less good for Proton showers where the separation of the energy is correct only up to 10%.

A more intuitive view of the test is given in Fig. 8 and 9. The points with error bars are the median and standard deviation of the control sample $E(G)$; the isolated points are the potentially falsified showers. The combination of criteria $Nmuons$ and $Nmax$ separates showers of different kinds. The dispersion of the $Nmax$ for protons for the two samples at 10^{19} and 10^{20} shows that the separation of energy cannot be improved on this criterion only.

More generally, a test scalable both in power (false positive) and significance (false negative) is probably not realistic, because of the intrinsic statistical variation of simulations; therefore, the test against a sample of control cannot be the only method for individual checks. On the other hand, accepting a relatively high number of false positive, allows to define a test with as good a significance as simulation allows for. Acceptance becomes then only an indication, which must be confirmed by full re-execution. This strategy is effective, because it actually rejects falsified showers at low cost. Let k be the ratio of execution time of G over $\mathcal{S}(G)$; the benefit of the test in comparison with re-execution is $1 - \alpha - k$; for instance, a 30% false positive rate and a 20% time ratio provide a 50% speedup.

6. CONCLUSION

This paper has presented a checking architecture focused on Monte-Carlo applications. The cost model of the test takes into account not only the abstract complexity - number of calls to an oracle - but also the resource management issues. Previous work on checking for global computing was more focused on test design than on overhead estimation. Our future work will explore result-checking of non one-way invertible functions through data mining.

Credibility appears as one variant of what has been considered as a major issue for large scale distributed systems: recovery-oriented computing [4]. While there exist an immense body of literature about protecting a distributed computation, the new vision of ROC is coping with failures, which translates to attacks in our context. The work described in this paper is also closer to the ROC strategy than for instance self-stabilizing algorithms: the user application remains unmodified; the middleware is in charge of taking the necessary actions, which are furthermore essentially binary -acceptance or rejection -, also in conformity with ROC principles.

7. ACKNOWLEDGMENTS

This work was partially funded by the *ACI GRID* programme of the french ministry of research.

8. REFERENCES

- [1] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley and Sons, 84.
- [2] M. Blum and S. Kannan. Designing programs that check their work. In *21st ACM Symposium on the Theory of Computing (STOC)*, pages 86–97, 1989.
- [3] M. Blum and H. Wasserman. Software reliability via run-time result-checking. *Journal of the ACM*, 44(6):826–849, 97.
- [4] A. Brown and D. A. Patterson. Embracing Failure: A Case for Recovery-Oriented Computing (ROC). In

- High Performance Transaction Processing Symposium*, 2001.
- [5] P. Buncic, A. J. Peters, and P. Saiz. The AliEn system, status and perspectives. In *Computing in High Energy and Nuclear Physics*. econf - <http://www.slac.stanford.edu/econf/C0303241>, 2003.
- [6] B. Falissard. *Les analyses intermdiaires dans les essais thrapeutiques*. PhD thesis, Universit Paris-Sud, 90.
- [7] G. Fedak, C. Germain, V. Neri, and F. Cappello. XtremWeb: A Generic Global Computing Platform. In *IEEE/ACM Int. Symp. Cluster Computing and the Grid (CCGRID'2001), Special Session Global Computing on Personal Devices*, pages 582–587, Brisbane, 2001. IEEE Press.
- [8] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *2nd Intl Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2573 of *LNCS*, pages 118–128. Springer-Verlag, 2003.
- [9] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *Computer*, 35(6):37–46, 2002. extended version: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, GGF June 2002.
- [10] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Intl Jal Supercomputer Applications*, 15(3):200–222, 2001.
- [11] C. Germain, G. Fedak, V. Neri, and F. Cappello. Global Computing Systems. In *3rd Int. Conf. on Large Scale Scientific Computations*, LNCS 2179, pages 218–227, Sozopol, 2001. Springer-Verlag.
- [12] C. Germain and N. Playez. Result-Checking in Global Computing Systems. In *Procs. 17th ACM Int. Conf. on Supercomputing*, pages 226–233, San Francisco, June 2003. ACM Press.
- [13] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Jal. ACM*, 45(4):653–750, 1998.
- [14] P. Golle and I. Mironov. Uncheatable distributed computations. In *Proc. of the RSA Conference 2001*, Lecture Notes in Computer Science, 2001.
- [15] Trusted Computing Group. <https://www.trustedcomputinggroup.org/downloads/specifications/>.
- [16] D. Heck and al. The Corsika program. Technical report, Forschungszentrum Karlsruhe, 98.
- [17] J. H. Hine and P. Dagger. Securing distributed computing against the hostile host. In *Procs of the 27th conference on Australasian computer science*, pages 279–286, 2004.
- [18] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky. SETI@home-Massively Distributed Computing for SETI. *Computing in Science and Engineering*, 3(1):78–83, 2001.
- [19] L. F. G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.
- [20] S. J. Sciutto. Aires : Air showers extended simulation. Department of Physics of the Universidad Nacional de La Plata, Argentina, 1995. <http://www.fisica.unlp.edu.ar/auger/aires/>.
- [21] D. Siegmund. *Sequential Analysis*. Springer Series in Statistics. Springer Verlag, 85.
- [22] D. Szajda, B. Lawson, and J. Owen. Hardening functions for large-scale distributed computations. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003.
- [23] A. Wald. *Sequential Analysis*. Wiley Series in Stat. Wiley, 66.

APPENDIX

This section recalls some results from [23].

$$L(p) = \frac{\left(\frac{1-\beta}{\alpha}\right)^h - 1}{\left(\frac{1-\beta}{\alpha}\right)^h - \left(\frac{\beta}{1-\alpha}\right)^h}$$

where h is defined by

$$p = \frac{1 - \left(\frac{1-p_1}{1-p_0}\right)^h}{\left(\frac{p_1}{p_0}\right)^h - \left(\frac{1-p_1}{1-p_0}\right)^h}$$

The values of h for $p = 0, p_0, p_1, 1$ are $+\infty, 1, -1$ et $-\infty$
Moreover

$$E_p(n) = \frac{L(p)\log B + (1 - L(p))\log A}{p\log\frac{p_1}{p_0} + (1 - p)\log\frac{1-p_1}{1-p_0}} \quad (3)$$

with $A = \frac{1-\beta}{\alpha}$ et $B = \frac{\beta}{1-\alpha}$