Fault-tolerant Distributed Computing Scheme based on Erasure Codes

Jérôme Lacan

ENSICA
1, place E. Blouin, 31053, Toulouse cedex France
jerome.lacan@ensica.fr

RÉSUMÉ. *De nouveaux types de systèmes de calculs répartis tels que les réseaux pair-à-pair ou les grilles de calculs sont composés de ressources de calculs hétérogènes dont la fiabilité est également fortement variable. Le travail présenté dans ce papier propose de transposer des codes correcteurs à effacement dans le domaine des processus pour améliorer la tolérance aux fautes des systèmes de calculs répartis. La méthode présentée consiste à générer des processus redondants à partir d'un ensemble de processus. Ce système permet de récupérer le résultat des calculs réalisés par l'ensemble des processus d'origine et redondants s'exécutant en parallèle même si certains d'entre eux connaissent des défaillances par arrêt.*

ABSTRACT. *Some emerging classes of distributed computing systems, such peer-to-peer or grid computing computing systems, are composed of heterogeneous computing resources potentially unreliable. This paper proposes to use erasure codes to improve the fault-tolerance of parallel distributed computing applications in this context. A general method to generate redundant processes from a set of parallel processes is presented. This scheme allows the recovery of the result of the application even if some of the processes crash.*

MOTS-CLÉS : *tolérance aux fautes, calculs répartis, codes à effacement, processus redondants*

KEYWORDS: *Fault tolerance, distributed computing, erasure codes, redundant processes*

1. Introduction

An increasing number of applications are based on distributed algorithms. These applications can use an heterogeneous network of computing resources with highly variable topology and characteristics. The improvement of the fault-tolerance of the distributed applications is a widely studied problem in this context [CAS 02]. Error-masking techniques are one of the classical approaches to enhance the fault-tolerance of distributed applications. In this paper, we introduce an error-masking technique consisting in adapting the IDA system of Rabin [RAB 89], which disseminates redundant data to improve the data availability in distributed storage systems, to distributed computing applications. With such a solution, the data will be available despite the crash of some of the processes/hosts.

This paper presents a method to use similar error correcting codes techniques to protect a set of distributed processes. The idea consists in defining a set of redundant processes from a set of given processes so that the failure of some of the given processes can be compensated by the redundant processes. The main problem we address here is how to define redundant processes from a set of processes.

Several powerful error correcting codes perform encoding and decoding operations with only simple `XOR` operations. The first step of our work consists then in defining a condition on an application in order to "mimic" the `XOR` operation. Intuitively, if a bit b_3 is obtained from two bits b_1 and b_2 with the `XOR` operation (i.e. $b_3 = b_1$ `XOR` b_2), it is possible to recover the bit b_1 (resp. b_2) from b_2 and b_3 (resp. b_1 and b_3) with the same operation. Let us consider now a distributed application partitioned into 2 processes $\mathtt{P}_1$ and $\mathtt{P}_2$ with inputs i_1 and i_2 and outputs o_1 and o_2. Let us assume that $\mathtt{P}_1$ and $\mathtt{P}_2$ are two instances of a same sub-application `P`. To reproduce the `XOR` scheme, we must define an operation on the inputs i_1 and i_2, called $X_{in}(i_1, i_2)$, so that the output o_1 (resp. o_2) can be recovered from o_2 and $\mathtt{P}(X_{in}(i_1, i_2))$ (resp. o_1 and $\mathtt{P}(X_{in}(i_1, i_2))$). The operation performing this recovery is called X_{out}. The applications supporting this scheme are called `XOR-able` applications.

The main idea is presented in the next Section. In Section 3, the general condition allowing the definition of the concept of `XOR-able` application is presented and several examples of such applications are presented. The classes of error correcting codes usable in this context are presented in Section 4. Section 5 concludes.

2. Error correcting codes and distributed applications

2.1. *Related work*

Few work have made the connection between Error correcting codes and distributed or parallel applications.

The Algorithm-Based Fault-Tolerance (ABFT) techniques [HUA 84] was introduced for parallel processors by Huang and Abraham as a means of error protection for

parallel matrix operations. The idea was to introduce checksums to provide error detection and in some cases, error correction. This approach can be applied efficiently for matrix-vector multiplication and for decompositions QR and LU.

In the domain of the consensus problem, [MOS 01] proved that if there exists a "relation" between the outputs of the processes, then the consensus problem can be solved for these processes despite a crash of some of them. The connection with error correcting codes is established in [MOS 02].

2.2. *Context*

The principle of some emerging classes of distributed computing systems, such peer-to-peer or grid computing systems, is to connect a wide variety of computer types and computing resources to create vast "virtual" reservoirs of computers serving geographically widely separated users. These computing resources range from desktop workstations to massively parallel processors. However, the heterogeneity of the resources makes the performance of the global system highly variable. Classically, this fault-tolerance is ensured by replicating the processes. Several general replication-based strategies were classified to be adapted to the characteristics of the applications and the distributed systems [POW 94].

A distributed computing system has many similarities with distributed storage systems. Indeed, the problem of the availability of the hosts also arises in distributed storage systems. It can be solved by using data dissemination schemes based on error correcting codes. The idea, proposed by Rabin [RAB 89], consists in splitting the data into k blocks which are encoded with an erasure code (i.e. an error correcting code designed for the erasure channel) to produce a set of n blocks disseminated on n hosts. If the erasure code is optimal, the data is available as long as k hosts among the n ones are alive.

The question of the applicability of such a scheme in distributed computing systems is then asked. In fact, a distributed computing system can be considered as a system providing input data to remote hosts and receiving output data from these hosts. With this definition, a distributed storage system is simply a distributed computing system where, on each host, the input data are equal to the output data. A natural way to generalize the fault-tolerant scheme is then to define a code such that 1) the encoding operation processes a vector of input data to generate redundant input data 2) the distributed processors process input data to produce output data 3)the decoding operation processes output data to produce the output data corresponding to the initial input data. Such a scheme is illustrated in Figure 1. However, compared to classical utilizations of error correcting codes, the problematic differs in several points. For example, by definition, the data are modified by the processes (input and output data differ). Moreover, the set of distributed applications can not be specified in terms of input and output alphabets (which can be different). The set of applications supporting such a scheme is formally defined in Section 3.

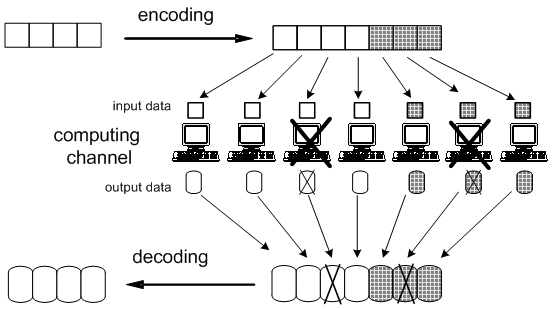

Figure 1. *Fault-tolerant scheme in distributed computing systems*

3. XOR-able applications

To define a general strategy to apply error correcting codes techniques to a maximum of applications, we propose in this Section to define an XOR-like operation for processes. Note that the goal of this definition is to propose a general but simple method to define redundant processes. It is clear that this definition can be generalized to applications which does not verify all the hypotheses. Let A be an application partitioned into k independent processes $P_1, \ldots, P_k$. We assume that all the processes perform the same operation, denoted by P, from a set of inputs I to a set of outputs O. Therefore, for any $1 \leq t \leq k$, each process $\mathtt{P}_t$ produces the output $o_t = \mathtt{P}(i_t)$ from the input i_t). The output of the application A is then the vector $(o_1, \ldots, o_k)$.

Definition 1 *The application A is XOR-able if there exists a function :*

$$X_{in} : I^k \longrightarrow I$$

and a set of k functions $X_{out}^{(1)}, \ldots, X_{out}^{(k)}$ *:*

$$X_{out}^{(t)} : O^k \longrightarrow O \quad \textit{such that for any } 1 \leq t \leq k,$$

$$X_{out}^{(t)}(o_1, \ldots, o_{t-1}, \mathtt{P}(X_{in}(i_1, \ldots, i_k)), o_{t+1}, o_k) = o_t.$$

X_{in} denotes the function processing input data and X_{out} denotes the function processing output data.

A first class of XOR-able applications are the applications such that P is a linear application. Indeed, let us consider an application A partitioned into k processes $\mathtt{P}_t$:

$$\mathtt{P}_t : \left\{ \begin{array}{ccc} I & \longrightarrow & O \\ i_t & \longrightarrow & o_t = \mathtt{P}(i_t) \end{array} \right.$$

such that $\mathtt{P}(a.i + b.i') = a.\mathtt{P}(i) + b.\mathtt{P}(i')$, where a and b are any element of a given field and where i and i' are any element of I.

Proposition 1 *The linear application* `A` *defined above is* `XOR-able`. X_{in} *and* $X_{out}^{(t)}$, *for* $1 \leq t \leq k$, *can be defined as follows :*

$$\left| \begin{array}{ll} X_{in} : & (i_1, \ldots, i_k) \longrightarrow X_{in}(i_1, \ldots, i_k) = \sum_{t=1\ldots,k} i_t \text{ and} \\ X_{out}^{(t)} : & (o_1, \ldots, o_{t-1}, \mathtt{P}(X_{in}(i_1, \ldots, i_k)), o_{t+1}, \ldots, o_k) \longrightarrow \\ & \qquad \mathtt{P}(X_{in}(i_1, \ldots, i_k)) - \sum_{u=1,\ldots,k;u\neq t} o_u \end{array} \right.$$

This proposition can be easily proved by considering that $\mathtt{P}(X_{in}(i_1, \ldots, i_k))$ is equal to $\sum_{t=1\ldots,k} P(i_t)$ since P is a linear application. Moreover, by definition, $o_t = P(i_t)$ for $t = 1, \ldots, k$. It follows that $\mathtt{P}(X_{in}(i_1, \ldots, i_k)) - \sum_{u=1,\ldots,k;u\neq t} o_u$ is equal to o_t.

A famous example of distributed linear applications is [KOR 01] where the main task of the distributed processes consists in performing Discrete Fourier Transforms, which are linear applications. Several more complex applications, e.g. related to linear algebra, can be considered as `XOR-able` applications (e.g. computing the inverse of k nonsingular square matrices, computing the determinant of several nonsingular matrices). Another class of `XOR-able` applications are some classical cryptographic operations such the operation $P(x) = g^x \% p$. Actually, even if the way to express a given distributed application as an `XOR-able` application is not always direct, it appears that this technique can be applied to numerous applications (or sometimes on some parts of these applications).

4. Error correcting codes for XOR-able applications

We consider here that the distributed processes crash or provide a correct result. The corresponding channel in terms of error correction is then the erasure channel. By construction, the `XOR-able` applications reproduce the scheme of the **Single Parity Check (SPC) codes** which protects k symbols with a additional symbol defined as the opposite of the sum of the k first symbols. It follows that this codes and all the codes derived from SPC can be used here. For example, a product code of several SPC, which reaches a high level of performance in term of correction capability (see [KOU 02]), can be direcly used to protect `XOR-able` applications. Note that these codes are able to correct more than one error per block.

More generally, any binary linear code whose the decoding only uses binary sums can be used for Several classical classes of binary linear error correcting codes can be used in this context. Reed-Muller, LDPC and "rateless" codes [LUB 02] belong to this class of codes.

5. Conclusion

This paper has presented a fault-tolerant distributed computing scheme based on error correcting codes. This scheme allows supporting the crash of some nodes. To apply error correcting codes concepts to processes, the notion of `XOR-able` applica-

tion was introduced and several examples of `XOR-able` applications were presented. Several classical error correcting codes were proposed to be used in this context.

Several problems are opened by this work. The first one is to verify whether it is possible to apply the proposed fault-tolerant scheme to given distributed applications such that, for example, classical matrix operations. Another problem concerns error correcting codes and more precisely the construction of efficient codes for distributed computing. Finally, the implementation of the fault-tolerant scheme in distributed systems opens several problems related to the consensus, the distributed decoding strategy or the dynamic management of redundant computing nodes.

Acknowledgments

The author thanks Tanguy Pérennou, Jérôme Fimes, Michel Salaün, *et al.* for their comments and suggestions.

6. Bibliographie

[CAS 02] CASANOVA H., « Distributed Computing Research Issues in Grid Computing », *Quarterly Newsletter for the ACM Special Interest Group on Algorithms and Computation Theory (SIGACT News)*, vol. 33, n° 2, 2002.

[HUA 84] HUANG K. H., ABRAHAM J. A., « Algorithm-based fault tolerance for matrix operations », *IEEE Trans. on Computers*, vol. C-33, 1984, p. 518-528.

[KOR 01] KORPELA E., WERTHIMER D., ANDERSON D., COBB J., LEBOFSKY M., « SETI@home :Massively Distributed Computing for SETI », *Computing in Science and Engineering*, vol. 3, n° 1, 2001, p. 78-83.

[KOU 02] KOUSA M., « A novel approach for evaluating the performance of SPC product codes under erasure decoding », *IEEE Transactions on Communications*, vol. 50, n° 1, 2002, p. 7 - 11.

[LUB 02] LUBY M., « LT Codes », *IEEE Symposium on Foundations of Computer Science*, 2002, p. 271-280.

[MOS 01] MOSTEFAOUI A., RAJSBAUM S., RAYNAL M., « Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems », *ACM Symposium on Theory of Computing (STOC'01)*, ACM, jul. 2001, p. 153-162.

[MOS 02] MOSTEFAOUI A., RAJSBAUM S., RAYNAL M., « Asynchronous Interactive Consistency and its Relation with Error-Correcting Codes », *ACM Symposium on Principles of Distributed Computing (PODC-02)*, jul. 2002, page 253.

[POW 94] POWELLS D., « Distributed fault-tolerance - Lessons learnt from Delta 4 », *IEEE Micro.*, vol. 14, n° 1, 1994, p. 36-47.

[RAB 89] RABIN M. O., « Efficient dispersal of information for security, load balancing, and fault tolerance », *J. ACM*, vol. 36, n° 2, 1989, p. 335–348, ACM Press.