SCALABLE CONCURRENCY CONTROL IN A DYNAMIC MEMBERSHIP – EXPERIMENTAL RESULTS*

Augusto Ciuffoletti
INFN-CNAF, Via Berti Pichat,Bologna, Italy
augusto@di.unipi.it

Ali Asim
University of Paris Sud-XI,Paris, France
Ali.Asim@lri.fr

Abstract   We introduce a solution for a concurrency control problem which is frequently encountered in practice: in a dynamic system we want that the load on a centralized resource is uniformly distributed among users, offering a predictable performance as long as it is not overloaded. We propose an original solution based on probabilistic assumptions, and we comment early experimental results.

Keywords: performance stabilization, randomized diffusion, membership management

---

1.   Introduction

Concurrency control is one of the basic building blocks of a distributed system. In this paper we examine a simple yet significant concurrency control instance: each node of the system must be enabled to execute a *critical operation* at approximately regular intervals, and we want the lapse between two successive executions to be decoupled from the size of the system.

We may find several use cases for this kind of concurrency control, even when we restrict our scope to the management of a Grid infrastructure: for instance, access to a global resource registry (the kind of problems otherwise solved using Distributed Hash Tables [5]), or execution of bulk data transfers [13] from data repositories. As a general rule, all applications for which a globally available resource, whose capacity we expect to grow with system size, can take advantage of such kind of concurrency control.

We describe how token circulation, a technique that is traditionally used to enforce mutual exclusion, can be extended to cover this case of concurrency control, and present experimental results that prove its effectiveness.

2.   Multiple tokens randomly moving

The circulation of a unique token whose presence enables the execution of the *critical operation* is a traditional topic of applied and theoretical research [6, 12]. However, the circulation of a unique token does not guarantee that each node is enabled periodically when the system size chages, since the *return time*, the lapse between two successive visits of the token to a given node, clearly depends on the size of the system.

The presence of several tokens is a less investigated technique [9, 3] which is suitable for our problem statement. Considering a constant and uniform *token latency*, the lapse between the receive of a token and its forward the the next node, one solution consists in changing the number of tokens linearly with system size. This is a quite original approach, and we found no bibliography on this subject.

In order to apply such technique, we apparently need to know a local estimate of system size: this is a quite intriguing research problem [2]. We explore an alternative which consists in using the feedback that we obtain from the observed *return times*; using that feedback we locally control token generation or removal. We expect that this form of closed loop control stabilizes the number of tokens so that each node is visited by a token with the required frequency. Such technique relies on the assumption that *token latency*, the interval between two successive token passing operations, is a stationery process.

Notably, the token circulation technique makes reference to a topology (by default circular), that guides the token across the system. The maintenance of an overlay ring on the fabric of the distributed system is a well studied topic;

however, the extension of known solutions to systems of thousands of nodes with unreliable links is problematic.

An alternative, which is valid when considering as the underlying fabric a full mesh topology like the transport level of the Internet, is to use a random walk approach: at each step the token is forwarded to another node selected at random in the distributed system [7, 11].

Such a technique introduces a further problem, since each node needs to access the directory of system nodes to select the next destination for a token. For the sake of reliability, we exclude the presence of a centralized registry, but we assume that each node collects a significant fraction of node identifiers. The resulting topology approximates a full mesh, since each node will be adjacent only to the peers listed in the local directory. However, theoretical analysis of random diffusion processes [8, 10] ensures that in such case the random walk process is similar to the case of a full mesh.

The collection of a membership directory may introduce scalability problems: our solution consists in piggybacking to each token the notice of recent join and leave events. This limits the response to the *churm*, the process of join and leave events, of our membership.

The availability of a directory of the membership is also required to ensure some security: when a node receives a token from a peer, it should be able to check whether the sender is included in the membership. In our solution we envision the storage of authenticated public keys in the directory: keys are used to produce a signature of each received token, before it is forwarded to the next peer.

Summarizing, our solution will feature the following techniques:

- a feedback mechanism that regulates the number of tokens is used to ensure the frequency of token arrival events on each node

- tokens are forwarded randomly based on the local knowledge of the membership

- each token is signed using sender's certificate

- each token carries recent membership changes observed by the sender.

2.1 Regulation of the number of tokens

Each node measures the lapse from last token passing operation, and computes an Exponentially Weighted Moving Average (EWMA) using successive values. The EWMA returns a robust estimate of the average *interarrival time* of a token on a given node.

Comparing the EWMA with the desired *interarrival time* the node is able to trigger compensating actions, that have the effect of optimizing the number of tokens in the system:

- the observation of an interarrival time longer than $k_{max} * \overline{t_{wait}}$ reflects in the generation of a new token;

- the observation of an interarrival rate shorter than $\overline{t_{wait}}/k_{max}$ reflects in enqueuing the token until that time;

- token overflowing the capacity c of the queue are removed from the system

The token buffering in the input queue smooths the variation of the number of tokens in the system: simply removing early tokens has been observed to induce a slower convergence.

The convergence of the above rules to the desired interarrival time depends on the distribution of the interarrival times for the specific system. It is possible to prove that the system converges to the desired value if, when the number of tokens ensures the requested interarrival time, the two events of token generation and removal have identical probabilities.

The parameters k_{max} and c should be computed so to satisfy such a requirement, but this cannot be done analytically. Therefore we opt for a first hit approximation suitable for an asymmetric distribution of interarrival times, the right queue being significantly longer than the left one.

In our experiment we used $k_{max} = 3$ and $c = 2$: such parameters showed to be somewhat conservative, since average interarrival time is better than expected, at the expenses of the number of circulating tokens.

2.2   Local registry management

When a node passes the token to another node, it includes in the token a list of recently observed join and leave events. The capacity of this list is a system-wide constant: we used the value of 10. Each element in the list contains the identifier of a node, and its certificate. Let us follow step by step a token passing event, starting from the send event.

The node that forwards a token piggybacks to the token its signature, computed on the whole token using its personal certificate. Starting from the first retry, the node sends also its certificate and, after some unsuccessful retries, it selects another peer.

Upon receiving a token, the node first verifies whether the peer is in the local directory, and if the signature matches with the certificate. If either test fails, the token is silently discarded, assuming that a safe peer will resend the same token together with its certificate. If the token contains a certificate, this is verified using the public key of the certification authority, and the signature is checked. If either test fails, the token is silently discarded.

The final step consists in updating the registry with the updates recorded in the token, and merging the update list with the local update list. Once the

critical activity controlled by the token is complete, the token is forwarded to another peer.

A join event fits smoothly in the above schema. The joining node obtains two input parameters: one personal certificate released by a well known Certification Authority (CA), and the address of another node in the system.

Using this information the node generates a new token, indicating itself as a joining member in the update list. The token is signed using the certificate, and sent to the contact peer. It is to be noted that the token generation event is expected to induce the eventual removal of a token.

The leave operation is managed similarly: upon leaving, a node produces a token with a leave event recorded in the update list. Failure detection is managed likewise, although the protocol is resilient to the presence of crashed nodes in the membership. In both cases the certificate of the failed node is permanently invalidated: in case of restart the node will need a new certificate.

We assume the presence of unique CA, whose public key is available to all nodes in the system, in charge of producing the certificates. The interplay between the CA and the generic node falls outside the scope of this paper.

3.   Experimental results

We notice that our solution features many decisions points that are resolved using probabilistic assumptions: we were not able to assess its validity in theory, and we attempted a experimantal validation using a prototype implementation run on a testbed of 191 nodes.

The protocol implementation counts approximately 200 Perl lines. Since we were mostly interested in system dynamics, we opted for two simplifications: we did not implement authentication and the leave event.

The former simplification is meant to have a small impact on the significance of our protoytpe: in a separate paper [4] we address experimental results for an authenticated token passing protocol, and the impact of signing/checking is minimal

The prototype has been run on a testbed on the development infrastructure Grid5000 [1]. Grid5000 is a research project of the French Government. The aim of this project is to develop a Grid platform where researchers and scientists perform large scale experiments. Seventeen French national laboratories are involved in this research project. The major funding agencies for Grid5000 include, the *Ministére de l'Education, de la Jeunesse et de la Recherche*, ACI Grid, INRIA, CNRS and some universities in France.

Grid5000 provides a very configurable and controllable environment for the experiments. It also provides the necessary software tools for the reservation of resources, deployment of experiments, for the monitoring and collection of results. Users can tailor the environment according to the particular nature of
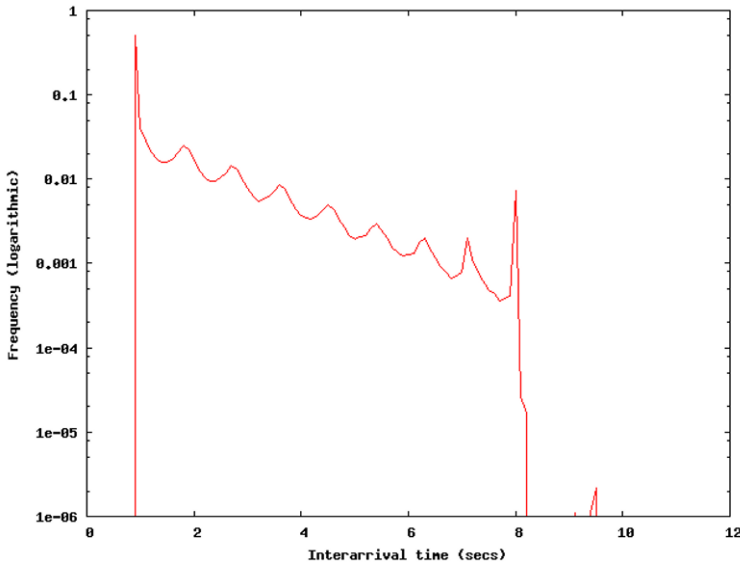
Figure 1. Frequency distribution of token interarrival time on nodes (log scale on y)

their experiments and data. To make the resource utilization optimal, the nodes on the Grid5000 can be shared between the users at the CPU and core level. Work is in progress to connect the Grid5000 with the Grid networks in other countries like Netherlands and Japan.

From the user viewpoint Grid5000 works much like a 70's batch system: a submitted job is run when resources are available. This fact motivated the adoption for our experiment of quite low time constants, in order to have a significant result after a minimum time. We opted for an expected *token latency* of 30 msecs, roughly corresponding to a token passing operation, thus obtaining very fast running tokens, and a target *token receive interarrival time* of 2.63 seconds. The size of the system was variable, and here we report an experiment with a number of nodes that gradually reaches 191 nodes. The duration of the experiment was set to 150 minutes: during the first 30 minutes the nodes grow from one to 191. With such settings, we want to observe the behavior of the system during a quite fast *power on* transient, and during a period of stability. Using an approximated model, not illustrated in this paper, we expect two tokens running in the system.

In order to assess the validity of our algorithm, we examine three quantities that give an idea of the dynamics of the system: the *token receive interarrival time* at each node, the number of tokens simultaneously circulating in the system, the time to diffuse the notice of a join to all nodes.

The distribution of the interarrival time is in figure 1. The plot shows a peak at the minimum value, corresponding to $\overline{t_{wait}}/k_{max} = 0.88secs$, indicating that tokens are usually enqueued before use: the "waving" trend of the plot has a similar reason.

The frequency of interarrival times drops exponentially (note the logarithmic scale on the y axis), and the probability of a interarrival time longer than 3 seconds is less than 1 percent: interarrival time shows an average of 1.71 seconds, instead of the target 2.63.

In figure 2 we see the dynamic variation of tokens during the experiment. It is quite evident the initial transient, during which the size of the system gradually grows from 0 to 191 nodes: the number of tokens follows a linear slope, and stabilizes to approximately 50 tokens, significantly more than expected.

Such a mismatch between the expected value and the experimental result is mainly due to the value of the parameters that control the token generation and removal rules, namely c and k_{max}: these parameters should be selected so that, when the number of tokens in the system is optimal for the target interarrival time, the probability of removing a token is equal to the probability of adding one. This fact would ensure that the number of tokens is in equilibrium when the number of tokens is optimal.

The evaluation of control parameters is difficult: although Internet performance is sufficiently uniform in time and space, the extremely randomized design of our token exchange protocol makes impossible to compute in advance the right value. An experimental tuning is therefore needed, but the robustness of the protocol ensures that the same values can be used in many different environments.

In our case, we used *first guess* values for our experiments: with the experience gained, we will refine the value of such parameters for next experiments.

Finally, we analyze figure 3 that helps evaluating the suitability of our protocol for broadcasting information to each node in the system. We recall that, for the correct operation of our protocol, we only need that most of the nodes, not necessarily all, receive a join update: the above results confirm this as a matter of fact, but leave a question mark on the time that is needed to inform *each* node about a join event.

Such results can be hardly checked against known theoretical results about the *cover time* of a random walk [10]: we recall that broadcasting is managed by a finite length list appended to the token, that several tokens are simultaneously running, and that each node merges its local event list with the one of the received token, before forwarding it.

Figure 3 shows that the broadcast time is quite long, with an average of 25 minutes. Depending on the application this time may be of interest.
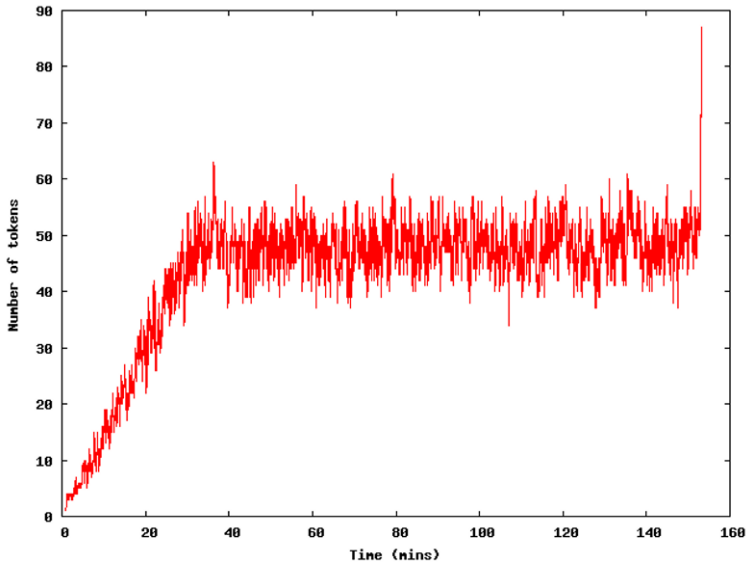
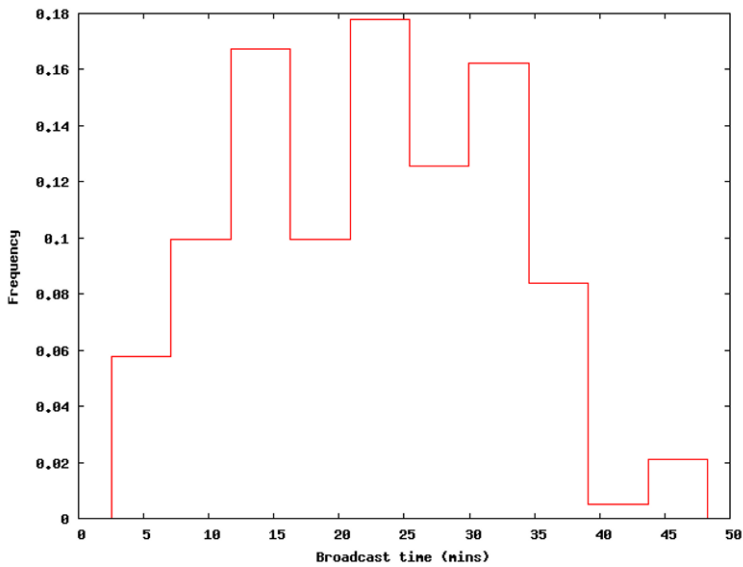*Figure 2.*   Variation of the number of tokens during the experiment

Figure 3.   Frequency of broadcast times for join events

4.   Conclusions and future works

The solution we propose makes extensive use of *probabilistic* assumptions: therefore many performance parameters are characterized by distributions, not

by deterministic values. In order to validate its operation we implemented a simplified version of the algorithm and we used Grid5000 as a testbed.

It is to be noted that the results in this paper come from one of the first experiments: more experiments are scheduled in the short term.

From the results available so far we understand that the algorithm, although probabilistic in nature, has a quite predictable behavior: the interarrival time of tokens is quite concentrated in time, its distribution falls exponentially, and is near to the requirements. An inside look at the engine shows that the number of tokens in fact follows the size of the system, with a quite fast response to variations. However, the number of tokens is higher than expected, and tokens are short lived: this is a matter for future investigation. Membership changes propagate and eventually reach all nodes in the system.

References

[1] Raphael Bolze, Franck Cappello, Eddy Caron, Michel Daydé, Frédéric Desprez, Emmanuel Jeannot, Yvon Jégou, Stephane Lantéri, Julien Leduc, Noredine Melab, Guillaume Mornet, Raymond Namyst, Pascale Primet, Benjamin Quetier, Olivier Richard, El-Ghazali Talbi, and Touche Iréa. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, November 2006.

[2] Javier Bustos-Jimenez, Nicolas Bersano, Elisa Schaeffer, Jose Miguel Piquer, Alexandru Iosup, and Augusto Ciuffoletti. Estimating the size of peer-to-peer networks using Lambert's W function. In *Proceedings of the CoreGRID Integration Workshop 2008*, pages 51–62, Hersonissos, Greece, April 2008.

[3] Nimmagadda Chalamaiah and Badrinath Ramamurthy. Multiple token distributed loop local area networks: Analysis. In *5th International Conference On High Performance Computing*, pages 400 – 407, December 1998.

[4] Augusto Ciuffoletti. Secure token passing at application level. In *1st International Workshop on Security Trust and Privacy in Grid Systems*, page 6, Nice, September 2007. submitted to FGCS through GRID-STP.

[5] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[6] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[7] Schlomi Dolev, Elad Schiller, and Jennifer Welch. Random walk for self-stabilizing group communication in ad-hoc networks. In *Proceedings of the 21st Simposium on Reliable Distributed Systems (SRDS)*, Osaka, Japan, October 2002.

[8] Feige. A tight lower bound on the cover time for random walks on graphs. *RSA: Random Structures and Algorithms*, 6, 1995.

[9] M. Flatebo, A.K. Datta, and A.A. Schoone. Self-stabilizing multi-token rings. *Distributed Computing*, 8(3):133–142, 1995.

[10] L. Lovasz. Random walks on graphs: a survey. In D. Miklos, V. T. Sos, and T. Szonyi, editors, *Combinatorics, Paul Erdos is Eigthy*, volume II. J. Bolyai Math. Society, 1993.

[11] Bernard Thibault, Alain Bui, and Olivier Flauzac. Topological adaptability for the distributed token circulation paradigm in faulty environment. In Jiannong Cao, editor, *Second International Symposium on Parallel and Distributed Processing and Applications – Hong Kong (China)*, number 3358 in Lecture Notes in Computer Science, pages 146–155. Springer, 2004.

[12] S. Tixeuil and J. Beauquier. Self-stabilizing token ring. In *Proceedings of the Eleventh International Conference on System Engineering (ICSE'96)*, Las Vegas, USA, July 1996.

[13] B. Volckaert, P. Thysebaert, M. De Leenheer, F. F. De Turck, B. Dhoedt, and P. Demeester. Network aware scheduling in grids. In *Proc. of the 9th European Conference on Networks and Optical Comm unications*, page 9, Eindhoven, The Netherlands, Jun 2004.