Poster: Self-tuning Batching in Total Order Broadcast via Analytical Modelling and Reinforcement Learning [*]

Paolo Romano
INESC-ID, Lisbon, Portugal

Matteo Leonetti
DIS, Rome University “Sapienza”, Italy

1. INTRODUCTION

Total order broadcast [2] (TOB) is a fundamental problem in distributed computing, which requires a set of processes to reach agreement on the delivery order of concurrently broadcast messages. Batching is a well known technique that allows boosting the throughput of Total Order Broadcast (TOB) protocols by amortizing the per-message ordering overhead across a set of incoming messages. Unfortunately, the manual configuration of the optimal batching level is a time consuming and delicate process, as incorrect tuning can lead to severe performance degradation.

In this paper, we overview an innovative mechanism for self-tuning the batching level of TOB protocols (a detailed description of which can be found in [3]), which combines analytical modeling and Reinforcement Learning (RL) techniques, to take the best of the two worlds: minimizing learning time and accumulating feedback from the operation of the system to enhance the self-tuning accuracy over time.

2. THE SELF-TUNING MECHANISM

We model analytically the node that sequences the messages (typically called sequencer) via a $M/M/1$ queue whose jobs are batches of size b. We denote with $\lambda(b,m)$, respectively $\mu(b,m)$, the arrival, respectively sequencing, rate of a batch of b messages, when the message arrival rate is equal to m. We express $\lambda(b,m)$ simply as m/b, and $\mu(b)$ as:

$$\mu(b,m) = \frac{1}{T_{1st} + \frac{(b-1)}{2m} + T_{add}(b-1)}$$

By using classic queuing theory results, we can then derive the TOB latency (as a function of m and b) as the response time of a M/M/1 queue.

As already mentioned, in order to compensate for the errors of the analytical model, we use a RL technique to update the knowledge of the model based on the feedback gathered during the operation of the system. We cast the problem of deciding the optimal batching level to a classical RL problem: the multi-armed bandit [1]. In this problem, a gambling agent is faced with a bandit (a slot machine) with k arms, each associated with an unknown reward distribution. The gambler iteratively plays one arm per round and observes the associated reward, adapting its strategy in order to maximize the average reward.

Our system relies on the UCB algorithm [1], a recent solution to the multi-armed bandit problem that provides logarithmic bounds on the number of suboptimal choices of the agent. We discretize the parameters space, given by the Cartesian product $b \times m$, and create for each message arrival rate interval an instance of the bandit with an arm for each discretized batching level. Further, we define a function to map the observed delivery latency to rewards distributed in the range [0,1] (as required by UCB). We combine UCB and the analytical model by initializing the statistics of every arm of each UCB instance with the corresponding latency predicted by the analytical model.

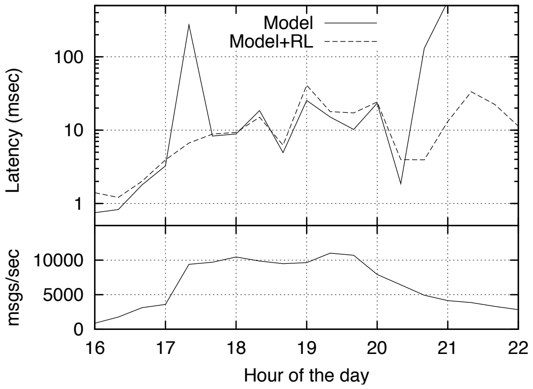

Figure 1: Evaluation of the self-tuning scheme

Our experimental results (see Fig. 1) highlight that, thanks to the initial knowledge of the analytical model, UCB avoids blind explorations of inadequately low batching values which would otherwise rapidly lead the system to trashing at high load. On the other hand, the usage of RL makes the self-tuning policy more accurate than when using exclusively the analytical model, whose initial, and partially incorrect, knowledge can be quickly updated by using UCB.

3. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[2] X. Defago, A. Schiper, and P. Urban. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4):372–421, 2004.

[3] P. Romano and M. Leonetti. Self-tuning batching in total order broadcast protocols via analytical modelling and reinforcement learning. Technical Report 9, INESC-ID, February 2011.

[*] This work was supported by the EU project “Cloud-TM” and by FCT (INESC-ID multiannual funding) through the PIDDAC Program Funds and the ARISTOS project.