Scientific Report for STSM project

Diego Didona
didona@gsd.inesc-id.pt

March 12, 2012

Reference Number	COST-STSM-IC1001-9454
Grant Number	COST-STSM-ECOST-STSM-IC1001-160112-013116
COST Action	IC1001
Action Description	Maximum degree of concurrency in distributed transactional memory platforms
STSM Period	2012-01-16 to 2012-02-16
Participant	Diego Didona Instituto Superior Técnico Distributed System Groups, INESC-ID
Host	Prof. Pascal Felber Université de Neuchâtel Institut D'Informatique

1 Purpose of the STSM

Transactional memory (TM) [6] has been widely studied over the last decade as it provides a scalable and easy-to-use alternative to locks. Over the last years, a wide body of literature has been published on TM, and several variants have been developed, including hardware-based (HTM), software-based (STM), and distributed (DTM) [5]. One of the key results highlighted by existing research is that, independently of the nature of the synchronization scheme adopted by a TM platform, its actual performance is strongly workload dependent and affected by a number of complex, often intertwined factors (e.g. duration of transactions, level of data contention, ratio of update vs read-onlye transactions).

The motivation for this STSM is the belief that most workloads have a *natural degree of parallelism*, i.e., there is a workload-specific threshold below which adding more threads will improve transaction throughput, and over which new threads will not help and might even degrade performance because of higher contention and aborts rates, even if sufficiently many cores are available.

The STSM's focus is on the importance of adapting the concurrency level to the workload (which is referred to as elastic scaling) in various application settings.

Related problems have been already addressed in previous research. For instance, Felber et al. [4] tackled the problem of how to tune at run time the number of "locks" and their coverage of the whole address space in the TinySTM library[1]. Wang et al. [9] exploit machine learning techniques to select which STM implementation to adopt on the basis of the application workload. Ansari et al. [1] adapt the parallelism of STM applications in order to maintain the transaction abort rate under a predefined level. In the area of replicated relational databases, recent works [8, 7] have proposed mechanisms for supporting elastic scaling, namely automatically adapting, in face of varying workloads, the number of nodes the platform is deployed onto. So far, however, limited attention has been devoted to dynamically identifying the optimal degree of parallelism for a (D)TM platform, namely the degree of local (i.e. number of active threads) and possibly global (i.e. number of nodes in a DTM) concurrency that maximizes the throughput of complex (D)TM applications. The purpose of this STSM is to investigate this topic,

[1] http://tmware.org/tinystm

taking as a starting point experiences and results already obtained separately by the research teams from Lisbon and Neuchâtel.

2 Description of the work carried out during the STSM

The STSM has been focused on the analysis of experimental results regarding the concept of natural degree of parallelism of an application, obtained considering two extreme scenarios: a shared-memory system with a low-level STM library written in C, and a distributed system with a high-level DSTM infrastructure written in Java.

Results taken from the first system show that realistic benchmarks exhibit widely different performance depending on the degree of parallelism, and that adapting the number of threads at runtime can improve performance of some applications over any execution with a fixed number of threads. An

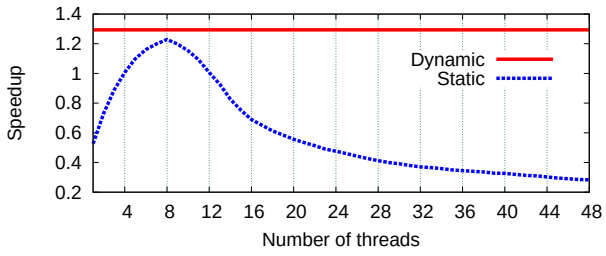


Figure 1: Speedup of the `intruder` benchmark as compared to sequential (non-STM) version, using static and dynamically evolving numbers of threads.

evidence of this is reported in Figure 1, where the benefits of adapting techniques are demonstrated on the application *intruder*, part of the widely used STAMP benchmark suite [2], when using TinySTM as transactional memory library. This application emulates a signature-based network intrusion detection system and exhibits a workload that evolves over time. By applying small modifications to the benchmarks and the underlying STM runtime in a shared-memory system, it is possible to optimize the concurrency level using exploration-based on-line optimization techniques, e.g., using hill climbing or gradient descent algorithms. Figure 1 indicates the performance of the benchmark when executed with varying number of threads (dashed line), or

when dynamically changing the number of threads (plain straight line corresponding to a constant value). It is easy to observe that performance is significantly better when dynamically adapting concurrency than with any fixed number of threads. It is important to remark that the experiment was run on a 48-core machine, i.e., the number of physical cores was not the limiting factor.

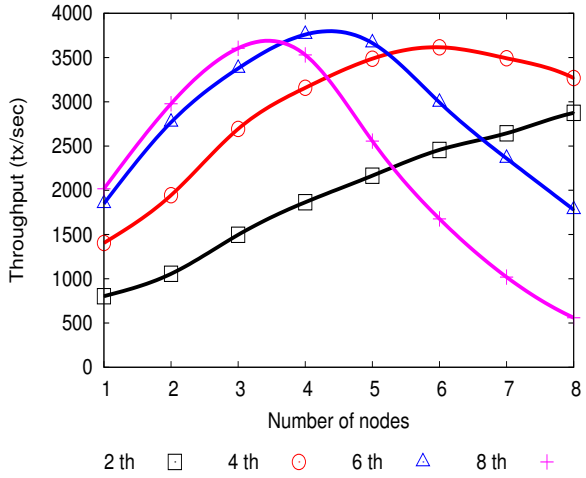
When considering DTM platforms [3], the problem of identifying the natural degree of parallelism for a given workload grows in complexity, turning into a bi-variate optimization problem. In distributed settings, in fact, the degree of concurrency is affected not only by the number of threads deployed on each node, but also by the number of nodes composing the distributed TM platform. The plots in Figure 2 clearly show the complex, non-linear interdependencies that can arise between the maximum throughput achievable by a DTM platform and both the number of nodes over which it is deployed and the number of threads running on every node.

The plots in Figure 2 were obtained by deploying and running the industry benchmark TPC-C[2] over a distributed system, using Infinispan[3] as distributed transactional memory library. As shown in the figure, there is a non-trivial correlation between throughput, number of nodes in the system and number of threads per node. The plots highlight that, when fixing the number of nodes, performance varies significantly depending on the amount of active threads on each node. This depends on the intertwining of effects associated with contention at both the data and physical level. As the number of nodes in the system increases, the latency of the DSTM's replication protocol grows accordingly, which, in turn, leads to an increase of the time threads spend idle waiting for the completion of replica synchronization activities. Clearly, a way to increase the utilization level of the cores available at each node is to activate additional threads on each node. However, this can lead to a raise of the level of data contention that can outweigh the performance gains achievable by pursuing higher levels of utilization of the available computational resources. As depicted by Figure 2b, this is precisely what happens in this case, as increasing the number of active threads per nodes in configurations with more than 4 nodes leads to thrashing due to the rapid growth of the transaction abort rate.

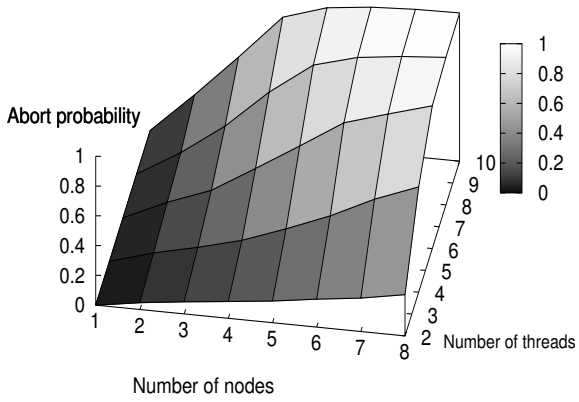
Unfortunately, in distributed settings it is impossible to exploit an explo-

[2]http://www.tpc.org/tpcc/

[3]www.jboss.org/infinispan



(a) Throughput (Committed Transactions per second)



(b) Transaction Abort Probability

Figure 2: Analyzing the performance of different configurations of the TPC-C benchmark when deployed on distributed TMs of variable sizes and local concurrency levels.

ration based approach, since the cost of testing configurations with a different number of threads (i.e., nodes) is prohibitive, as it requires transferring state, generates additional traffic, and takes orders of magnitude more time than in centralized settings. Therefore, in such settings, it is mandatory to rely on modeling techniques to predict the expected gains from adding or removing nodes for adapting the concurrency level.

The assessment of the modeling approach has been carried out by exploiting a tool developed by the Lisbon team, namely Transactional Auto Scaler (TAS); to the best of our knowledge, it is the only model aimed at driving elastic scaling decisions in DSTMs which takes into account both local and global degree of concurrency. In order to perform its prediction, TAS relies on analytical models to capture the concurrency control and replication schemes' behavior of the DSTM and on machine learning to obtain the value for the response time relevant to physical components (i.e., CPU and network).

The plots in Figure 3 compare the actual performance of the system with the performance predictions generated by TAS. To this end, TAS was provided with information concerning utilization of logical and physical resources while running in a configuration with 2 nodes and 2 threads, and was queried in order to obtain performance forecasts while varying the number of nodes and threads in the system. The plots highlight the ability of TAS to forecast correctly not only the performance trends for the considered workload as a function of the degree of concurrency in the system, but also the configuration that maximizes systems throughput. On the other hand, the plots also highlight the existence of system configurations in which the accuracy of the performance predictions output by TAS can decrease significantly. This depends on two main factors. The first one is that the analytical model employed in TAS for capturing the effect of data contention relies on assumptions whose validity can be challenged at extremely high levels of contention. This claim is confirmed by correlating the plots of Figure 3 with the abort probability reported in Figure 2b, and noting that the accuracy of TAS prediction degrades in regions where the systems throughput starts dropping in settings exhibiting extreme levels of data contention (higher than 80%); note that this loss of accuracy in such scenarios is natural for analytical models, as shown in[10]. The second factor that contributes to degrade the quality of TAS predictions is imputable to the loss of accuracy of its machine-learning based models. Off-line learners predictive power strongly depends on the representativeness of the samples observed during their training phase; however, the parameters space associated with all possible workloads and concurrency levels in a DTM is extremely vast, thus exploring it in an exhaustive fashion is prohibitive. Thus, in TAS we use a uniform sampling of the features space, and limit the duration of the off-line benchmarking to one hour. The downside of this approach is that it can lead to a degradation of the predictions accuracy especially in those regions in which small fluctuations of the input

variables lead to strong variations of the output variables (e.g., in case new threads are added in high CPU utilization scenarios).

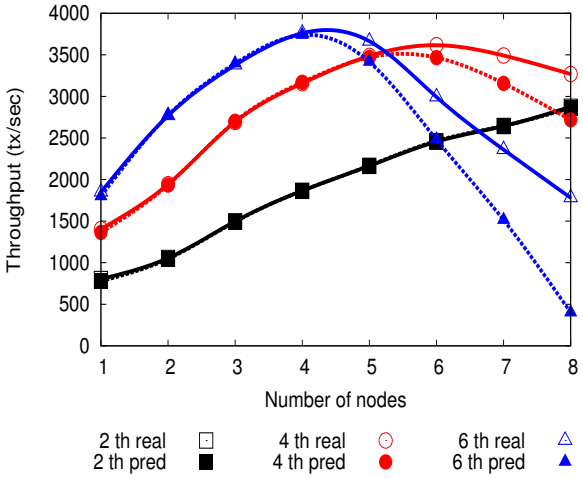


Figure 3: Accuracy of TAS performance predictions.

3 Description of the main results obtained

The main result obtained during the STSM is the production of a position paper, which has been submitted to an international workshop and is currently under review. In this paper we first report the results of analytical modeling and on-line exploration techniques; then we propose new methodologies to tackle the problem of determining the optimal level of parallelism for an application when deployed over a DSTM.

In the light of the experimental results, we observed that exploration-based techniques shine in determining the optimal degree of local concurrency, avoiding the usage of time-consuming off-line training phases, or the design and validation of analytical models, which are instead necessary for model-driven techniques. The other side of the coin is that approaches based exclusively on exploration-based techniques result cumbersome and unattractive to determine the optimum number of nodes to use in a DTM, due to the high overheads associated with state transfer. Model-based performance forecasting techniques, on the other hand, lend themselves better to tackle

this problem. However, their accuracy is ultimately dependent on the quality of the models they employ, which can be affected by several factors, as pointed out in the previous section.

In the paper, we advocate the joint usage of these two methodologies, claiming that on-line exploration can improve accuracy of models, while models can improve scalability of on-line techniques. More in detail, we have identified three alternative techniques in which modeling and on-line exploration are combined with the purpose of defining an accurate and robust methodology to drive elastic scaling decision in DSTM.

- Using model-based techniques to initialize the knowledge base of on-line learning mechanisms. This would allow to guide their exploration in order to avoid dwelling in regions that are clearly identified as unfavorable by model-based oracles, or to adjust dynamically the granularity of the on-line research steps.

- Combining the two methodologies according to a divide-and-conquer fashion. With this approach, the problem of identifying the optimal degree of concurrency in a DTM platform would be decoupled into two simpler sub-problems (namely determining the optimal number of distributed nodes vs. the optimal number of threads active per node), which could then be solved using the most appropriate methodology. For instance, model-driven techniques, such as TAS, could trigger a global reconfiguration of the system, which could then be followed by exploration-based approaches aimed at ad- justing the local concurrency level at each node in order to compensate possible errors of the model.

- Incorporating the feedbacks gathered using on-line exploration techniques into performance forecasting models, with the ultimate goal of continuously enhancing their predictive power and overall robustness. Conventional white-box models of transactional platforms [5, 3, 19, 6], for instance, are built on rigid assumptions (e.g., exponential arrival rate of lock- requests) and are not conceived to adapt themselves to cope with scenarios in which such assumptions are not (or are only partially) met.

4 Foreseen publications resulting from the STSM

As already mentioned in Section 3, a paper describing the results collected during the STSM has been submitted to HotPar'12[4] and is currently under review. The title of the paper is *Elastic Scaling for Transactional Memory: From Centralized to Distributed Architectures*; the full list of authors is the following: Diego Didona, Pascal Felber, Derin Harmanci, Paolo Romano, Jorg Schenker.

5 Future collaboration with the host institution

Both research teams have already decided to continue to research together by investigating the viability of the approaches proposed in the submitted paper. The purpose of this further collaboration is building a robust tool which is able to determine at runtime the optimal level of parallelism for an application, both on a local and global scale. Work is being currently carried out remotely at the two institutions, with periodic checkpoints via email and Skype. We plan to apply for a second STSM, after having reached more mature research results, in order to finalize the preparation of a joint paper.

6 Confirmation by the host institution of the successful execution of the STSM

Prof. Pascal Felber acknowledges the successful completion of the STSM, as certified by the attached document Confirmation.pdf.

[4] http://static.usenix.org/events/hotpar12/index.html

References

[1] Ansari, M., Luján, M., Kotselidis, C., Jarvis, K., Kirkham, C., and Watson, I. Robust adaptation to available parallelism in transactional memory applications. *Transactions on High Performance and Embedded Architectures and Compilers 3*, 4 (2008).

[2] Cao Minh, C., Chung, J., Kozyrakis, C., and Olukotun, K. STAMP: Stanford transactional applications for multi-processing. In *IISWC* (2008), pp. 35–46.

[3] Couceiro, M., Romano, P., Carvalho, N., and Rodrigues, L. D2stm: Dependable distributed software transactional memory. In *PRDC'09: Proceedings of the 15th Pacific Rim International Symposium on Dependable Computing* (Shanghai, China, Nov. 2009).

[4] Felber, P., Fetzer, C., and Riegel, T. Dynamic performance tuning of word-based software transactional memory. In *PPoPP 08: 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Feb. 2008), pp. 237–246.

[5] Harris, T., Larus, J. R., and Rajwar, R. *Transactional Memory*, 2nd edition ed. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publisher, 2010.

[6] Herlihy, M., and Moss, J. E. B. Transactional memory: Architectural support for lock-free data structures. In *ISCA* (1993), pp. 289–300.

[7] Qi Zhang, Ludmila Cherkasova, Evgenia Smirni. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Proceedings of the Fourth International Conference on Autonomic Computing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 27–.

[8] Saeed Ghanbari, Gokul Soundararajan, Jin Chen, Cristiana Amza. Adaptive learning of metric correlations for temperature-aware database provisioning. In *Proceedings of the Fourth International Conference on Autonomic Computing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 26–.

[9] Wang, Q., Kulkarni, S., Cavazos, J., and Spear, M. Towards applying machine learning to adaptive transactional memory. In *6th ACM SIGPLAN Workshop on Transactional Computing (TRANSACT)* (June 2011).

[10] Yu, P. S., Dias, D. M., and Lavenberg, S. S. On the analytical modeling of database concurrency control. *J. ACM 40* (1993).