CHAPTER TWO

TOWARDS AUTONOMIC TRANSACTIONAL REPLICATION FOR CLOUD ENVIRONMENTS

MARIA COUCEIRO, PAOLO ROMANO,
LUÍS RODRIGUES

In recent years, in-memory transactional data platforms, often referred to as NoSQL data grids, have emerged as a reference solution for transactional data management in the cloud. In these in-memory platforms, replication plays a role of paramount importance for fault-tolerance purposes, given that it represents the key means to ensure data durability in the face of unavoidable node failures. Unfortunately, despite the abundance of approaches in the area, existing replication schemes still fall short of addressing one of the key requirements of cloud computing environments: ensuring optimal efficiency when deployed over elastic infrastructures that dynamically expand/reduce the number of (physical or virtualised) resources in response to fluctuations of workload characteristics.

Tackling this issue is a key goal of the Cloud-TM project, a recent EU funded initiative that aims at developing a self-tuning, elastic transactional data platform tailored to meet the elasticity requirements of cloud computing infrastructures.

1. Introduction

By shifting data and computation away from local servers towards very large scale, world-wide data centres, cloud computing promises compelling benefits for both cloud consumers and cloud services providers (Armbrust et al. 2009). However, the promise of infinite scalability catalysing much of the recent hype about cloud computing is still menaced by one major pitfall: the lack of programming paradigms and abstractions

capable of bringing the power of parallel programming into the hands of ordinary programmers.

One of the most crucial issues to tackle when developing applications for the cloud (and in general for large scale distributed systems) is the management of concurrent access to a shared state. The challenge here is to identify mechanisms capable of ensuring adequate consistency levels while being:

(1) simple and familiar for the programmers

(2) highly efficient and scalable

(3) fault-tolerant and highly available

Cloud-TM (www.cloud-tm.eu) is an ongoing European Project that is tackling precisely these issues by developing a self-optimising middleware platform aimed at simplifying the development and administration of applications deployed on large scale cloud computing infrastructures.

At the core of the Cloud-TM platform lies the abstraction of a Distributed Software Transactional Memory (DSTM) (Carvalho et al. 2010; Couceiro et al. 2009). DSTMs are a recently proposed extension of the Transactional Memory (TM) (Shavit and Touitou 1997) programming paradigm, which was originally introduced to simplify the development of concurrent, though not distributed, programs. By releasing the programmer from the burden of managing locks or other error-prone low-level concurrency control mechanisms, TMs have been shown to enable a significant boost in productivity, to shorten development times, and to increase code reliability in complex concurrent applications. DSTMs have recently garnered significant interest as a more flexible and scalable alternative to conventional relational DBMSs (Stonebraker et al. 2007), particularly attractive in cloud environments. On the industrial side, the market of in-memory transactional data grids (as DSTMs are typically referred to in these enviroments) has undergone a rapid proliferation, and nowadays offers a good number of platforms both as proprietary products (e.g. Oracle© Coherence) and open source projects (e.g. JBoss© Infinispan). On the academy side, this area is attracting the interest of a growing community of researchers from areas such as distributed computing theory (Attiya et al.2010), database systems (Kallman et al. 2008), high performance computing (Bocchino et al. 2008) and, naturally, transactional memories (Herlihy and Sun 2007).

In these in-memory platforms, replication plays a role of paramount importance for fault-tolerance purposes, since it represents the key means to ensure data durability in the face of node failures. Unsurprisingly, the

replication schemes employed in these platforms take inspiration from the vast amount of literature on replication of transactional systems (Patino-Martínez et al. 2000, Couceiro et al. 2009). However, despite the abundance of existing replication strategies, no universal, one-size-fits-all solution exists providing optimal performance in highly heterogeneous/ fluctuating workloads and independently of the scale of the underlying platform.

The complexity of this problem is therefore exacerbated in cloud computing platforms due to the feature regarded as one of the key advantages of the cloud: its ability to elastically acquire or release resources, de facto dynamically varying the scale of the platform in real-time to meet the demands of varying workloads.

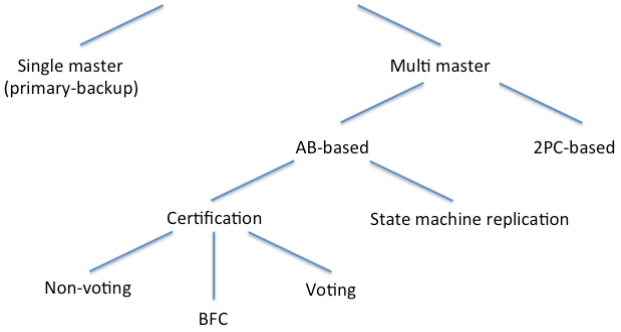
In light of these considerations, we advocate that, in order to match the elasticity requirements of cloud computing infrastructures, and to maximize efficiency, in-memory transactional data grids should entail multi-modal replication strategies and be able to dynamically self-tune their operating mode.

The structure of this chapter is as follows: first, in Section 2, by surveying existing approaches for state consistency in replicated transactional platforms, a taxonomy of the main solutions in literature is presented. In Section 3, we complement the critical analysis with a set of performance evaluation studies aimed at exposing advantages and drawbacks of existing solutions. Our results provide experimental evidence of the fact that fluctuations of the workload can have a strong impact on the performance of existing replication protocols, motivating the need for investigating self-tuning mechanisms. In Section 4, an case study highlighting the benefits achievable via the proposed self-tuning replication mechanism is illustrated. Finally, Section 5 presents an overview of the online research lines, shedding light on some of the key challenges that need to be addressed to achieve this ambitious goal, and providing pointers to recent results in this area.

2. A Taxonomy of Transactional Replication Protocols

Over the last decades, a vast amount of literature on replication of transactional systems has emerged (Pedone et al. 2003; Kemme and Alonso 1998; Patino-Martínez et al. 2000). The various solutions can be classified according to the following taxonomy in fig. 2.1.

Fig. 2.1. Taxonomy for full-replication transactional solutions



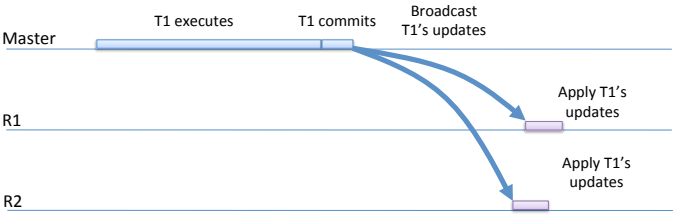
A first classification parameter is whether write transactions can be executed on a single node or by all nodes in the system: single versus multi-master schemes.

Single-master. In single master schemes, also known as primary backup, write transactions are executed exclusively at a single node (also called master or primary), whereas the remaining replicas can only run read-only transactions. Upon failure of the master, a backup replica is elected to become the new master.

Fig. 2.2 below exemplifies the execution of a transaction is a system with two backup replicas. The master executes transaction T1 and, after committing it, broadcasts its updates to the replicas, which apply them in order.

Note that, as the write transactions can be serialised locally by the master using its local concurrency control algorithm, this approach can rely on a simple replica synchronisation scheme with respect to multi-master solutions (as we will see shortly).

Fig. 2.2. Single-Master



On the down side, the throughput of write transactions does not clearly scale up with the number of nodes in the system, which makes the master prone to becoming the system bottleneck.

Multi-master. Multi-master schemes, on the other hand, are typically more scalable as transactions can be processed in all nodes. There are two types of synchronising the access to data: eager and lazy. The first relies on a remote synchronisation phase upon each (read/write) access, which normally results in very poor performance results (Franklin et al. 1997). Conversely, the lazy approach defers replica synchronisation until the commit time, which is when the transaction is finally validated. Lazy multi-master schemes can be classified based on whether they rely on Atomic Commit Protocols (such as Two-Phase Commit) or Atomic Broadcast (AB) (Defago et al. 2004) schemes to determine the global serialisation order of transactions.

Two-Phase Commit. In solutions based on Two-Phase Commit (2PC), transactions attempt to atomically acquire locks in all nodes. Even though they normally incur in minor communication overheads with respect to those relying on AB, these solutions are well known to suffer scalability problems due to the rapid growth of the distributed deadlock rate as the number of replicas in the system grows (Gray et al. 1996).

Fig. 2.3. The Two-Phase Commit protocol

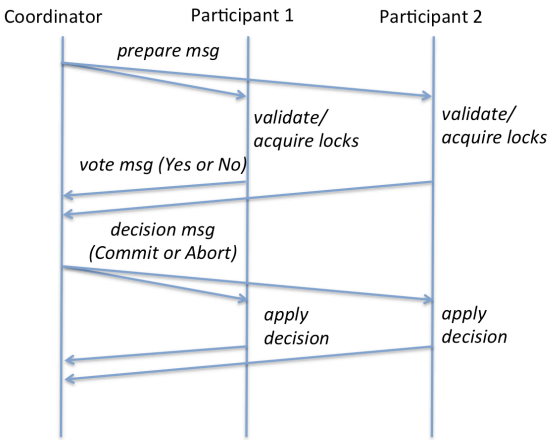


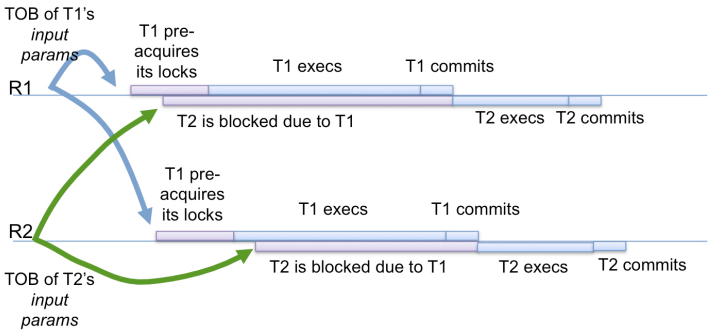
Fig. 2.3 above shows the 2PC protocol in action. The coordinator first sends the prepare message with the transaction. Both participants validate the transaction, acquire the locks of the objects read and written and reply to the coordinator with "commit" or "abort." If all participants answer commit, the coordinator sends a decision message with the indication of committing the transactions. If at least one participant replies abort, the coordinator sends an abort message. The participants apply the decision of the coordinator and the latter ends the protocols after receiving the acknowledgement from all participants.

Atomic Broadcast based replication. Conversely, AB-based replication is a family of (distributed) deadlock-free algorithms that serialise transactions in the total order established by an AB service (Defago et al. 2004). These solutions can be distinguished into two further classes: state machine replication and certification.

State Machine Replication. In the state machine replication (Schneider 1993), all replicas execute the same set of transactions in the same order. The transactions are shipped to all replicas using total order broadcast and, consequently, all replicas receive transactions in the same order and execute them in that order. However, both transactions and the validation scheme must be fully deterministic so that all replicas begin and end transactions in the same state.

Fig. 2.4 depicts the execution of the State Machine Replication in two replicas. Replicas 1 and 2 atomic broadcast the request to start transactions T1 and T2. T1 is ordered first—it (atomically) pre-acquires all the locks it requires, executes and commits in both replicas. T2 must wait until the locks are released to be executed also in both replicas.

Fig. 2.4. State Machine Replication



This solution has the key advantage of never incurring in transactional aborts. On the other hand, it requires a-priori knowledge of the transaction's read-set and write-set. Further, the execution cost of write transactions has to be fully paid by all the replicas in the system, which may limit the scalability of this approach in write dominated workloads.

Certification. Certification based techniques take a more optimistic approach and have been shown to achieve higher scalability by fully executing the transaction only at one node. This means that different transactions may be executed on different replicas concurrently. If the transaction aborts, no coordination is required. However, if the transaction is ready to commit, coordination will both ensure serialisability (Bernstein et al. 1986) and propagate the updates to the other replicas. This means that two transactions may update concurrently the same data in different replicas and it is up to the coordination phase to detect this situation and abort at least one of the transactions.

In this work, we will be mainly focusing on the following three certification protocols: non-voting (NVC), voting (VC) and Bloom filter (BFC) certification.

Non-Voting Certification. As with all certification protocols, in Non-Voting Certification (Pedone et al. 2003; Patino-Martínez et al. 2000) the transaction first executes locally. Then, when it is ready to commit, its read and write-set are sent to all replicas using Atomic Broadcast. Consequently, all replicas certify the transaction in total order. The validation phase consists of checking if the read-set is still valid or, in other words, if no other transaction has updated any item in the read-set. If the read-set is valid, the transaction can be committed. Otherwise, it is aborted.

Fig. 2.5. Non-Voting Certification protocol

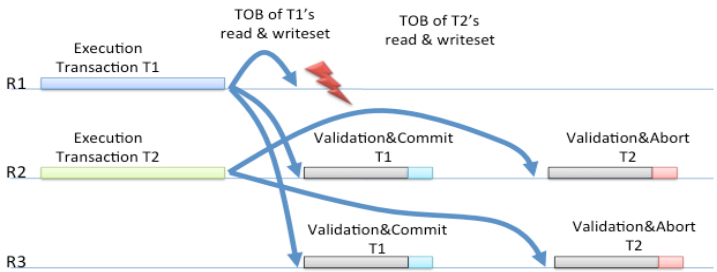


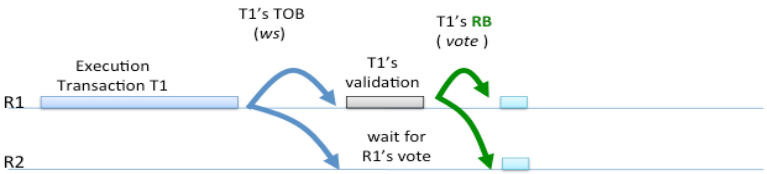
Fig. 2.5 above exemplifies the execution of NVC when one replica fails. Two transactions, T1 and T2, execute concurrently in two replicas. Atomic broadcast orders T1 before T2. R1 fails, but since the remaining replicas both received T1's read and write sets, they validate the transaction and apply the updates. T2 is then delivered to R2 and R3, validated and aborted due to conflicts with the already committed T1.

The key advantage of NVC consists of necessitating a single AB-based communication phase to determine the outcome of transactions. However, since the entire read-set must be sent together with the write set, messages exchanged by the replicas become very large, resulting in high levels of network congestion.

Voting Certification. The Voting Certification (Kemme and Alonso 2000) tries to overcome this issue by only ABcasting the write-set of locally executed transactions. As fig. 2.6 shows, commit requests are processed in total order but, in this case, only the replica where the transaction was executed can validate it since no other replica can access its read-set. When this replica finishes the validation, it sends the outcome to all replicas using reliable broadcast (Guerraoui and Rodrigues 2006), a lightweight communication primitive that guarantees all replicas receive the message. When this message is received, replicas either apply the write set of the corresponding transaction or discard it.

The messages exchanged by the replicas are significantly smaller compared to those generated by NVC. However, this protocol requires one additional communication step to disseminate the outcome of the transaction.

Fig. 2.6. Voting Certification protocol



Bloom Filter Certification. Bloom Filter Certification (Couceiro et al. 2009) consists of encoding the read-set of transaction in a Bloom Filter (Bloom 1970), a space-efficient data structure that strongly (and efficiently) compresses the messages disseminated via the AB service, while still allowing every replica in the system to deterministically certify

the transactions. Unlike voting schemes, BFC avoids additional communication steps during the commit phase. In terms of generated network traffic, even though BFC generates larger messages than voting protocols, it typically reduces the size of the messages exchanged via the AB service significantly when compared to non-voting schemes. On the down side, BFC can suffer from false positives due the probabilistic nature of Bloom filter-based encoding, which ultimately lead to an additional rate of aborted transactions.

3. No "One-Size-Fits-All" Solution

We can conclude from the previous section that the design space of distributed state consistency mechanisms is so vast that no universal, one-size-fits-all solution exists. The efficiency of individual state management approaches can in fact be strongly affected by both:

(1) the characteristics of the incoming workload, such as the ratio of read/write operations, as well as the spatial/temporal locality in the data access patterns, and

(2) the scale of the system (e.g. low vs high number of nodes, local vs geographical distribution) on which these mechanisms are deployed.

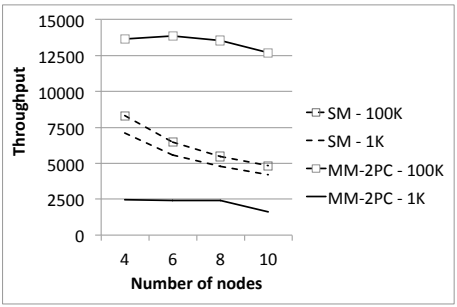
In order to support this claim, in the remainder of this section we present an experimental study to compare the performance of the previously presented protocols in distinct workload/scale settings.

The tests were run on a cluster of eight nodes, each one equipped with two Intel Quad-Core XEON at 2.0 GHz, 8 GB of RAM, running Linux 2.6.32-26-server and interconnected via a private Gigabit Ethernet.

In the two first experiments, the test system consisted of a key/value store implemented in Java. We used a synthetic benchmark and two different contention scenarios: the first consisted of a data set with 100,000 items, while in the second the data set had only 1,000 items. Note that by varying the size of the data set, the probability of contention among transactions also varies. Specifically, the contention probability is around two orders of magnitude larger when using the smaller data set. Each transaction performed ten reads and one update operation, choosing the data items to be accessed uniformly from the data set.

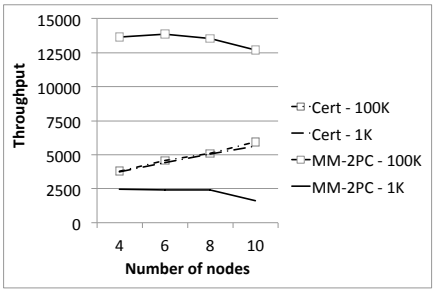
Single versus Multi-master. The graph in fig. 2.7 compares the throughput (number of transactions per second) of single-master (SM) and an implementation of multi-master using the 2PC protocol (MM-2PC) that is run at the end of each transaction. This experiment clearly showed that neither solution outperforms the other in both scenarios. For the low contention scenario, the MM-2PC is able to achieve a much higher throughput than single-master, whereas when there is high contention in the data set, the single-master scheme outperforms MM- 2PC. This is due to the fact that, while single-master is unaffected by the contention, the number of conflicts has a huge impact in the performance of MM-2PC, and the low throughput is the result of the high number of deadlocks.

Fig. 2.7. Throughput of single-master and multi-master 2-Phase Commit protocols



Multi-master versus Certification. The plot in fig. 2.8 compares the performance of MM-2PC and certification. The MM-2PC performance has the best performance for the low contention scenario. Certification outperforms MM-2PC in high contention scenarios once more due to the high rate of deadlocks in the latter.

Fig. 2.8. Throughput of Certification and multi-master 2-Phase Commit protocols



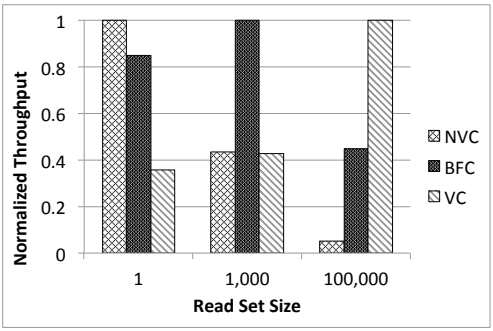
There are many approaches to implement Atomic Broadcast (Defago et al. 2004), for instance sequencer (one node is responsible for ordering the messages) or token (nodes can only send messages when they own a token). For this test we used a sequencer based implementation of Atomic Broadcast which limits the maximum throughput achievable by the system to the maximum number of messages that can be ordered by the sequencer. This is the only bottleneck in the performance of the protocol, as the contention level does not influence the throughput .

Non-Voting versus Bloom Filter versus Voting Certification. Following the discussion presented in Section 2, the performance of each of these three alternative certification mechanisms is strongly dependent on the actual distribution of the size of the read-sets generated by the transactional application.

Fig. 2.9 below shows the results of a sensitivity analysis aimed at assessing the actual impact of the read-set size distribution on the performance of the three previously described certification schemes. The results were obtained using a simple synthetic benchmark adapted from the Bank Benchmark originally used in Couceiro et al. (2009). Focusing only on the effects due to variations of the read-set size, which represents the goal of this sensitivity analysis, the benchmark was configured to never generate conflicts among transactions. The only aborts experienced in the system are therefore those determined by false positives with the BFC scheme (which was configured to have an additional abort-rate of 1%, as in Couceiro et al. 2009). The in-memory transactional data grid and the certification protocols were implemented in Java. The system uses two main components: (i) a state-of-the-art Software Transactional Memory (STM), namely JVSTM (Cachopo and Rito-Silva 2006), used to manage

local concurrency, and (ii) a replicated key/value store, used to maintain associations between unique object identifiers and object instances.

Fig. 2.9. Throughput of three certification strategies with different read-set sizes



Our experimental results highlight that no one-size-fits-all-solution exists that maximizes throughput across all the considered workloads. On the contrary, NVC provides the best performance in the scenario with small read-sets, BFC the best in the scenario with 1,000 items in the read-set, and VC is by far the best with large read-sets. Further, the relative difference in the performance between the best and worst performing protocol for each scenario ranges from a factor 2.5x (BFC vs VC, read-set size equal to 1,000) to 10x (VC vs NVC, read-set size equal to 100,000).

4. The Cloud-TM Approach

Section 3 provided a number of experimental evidences highlighting the benefits that could be achieved by dynamically adapting the replication mechanism in an in-memory transactional data platform. Fig. 2.10 below illustrates a possible example scenario highlighting how the autonomic, self-optimising capabilities that we are developing in the Cloud-TM platform will be exploited by user level applications.

The bottom part of the figure shows the variations over time of (a) the incoming traffic, (b) the ratio of read/write transactions, and (c) the transaction conflict probability.

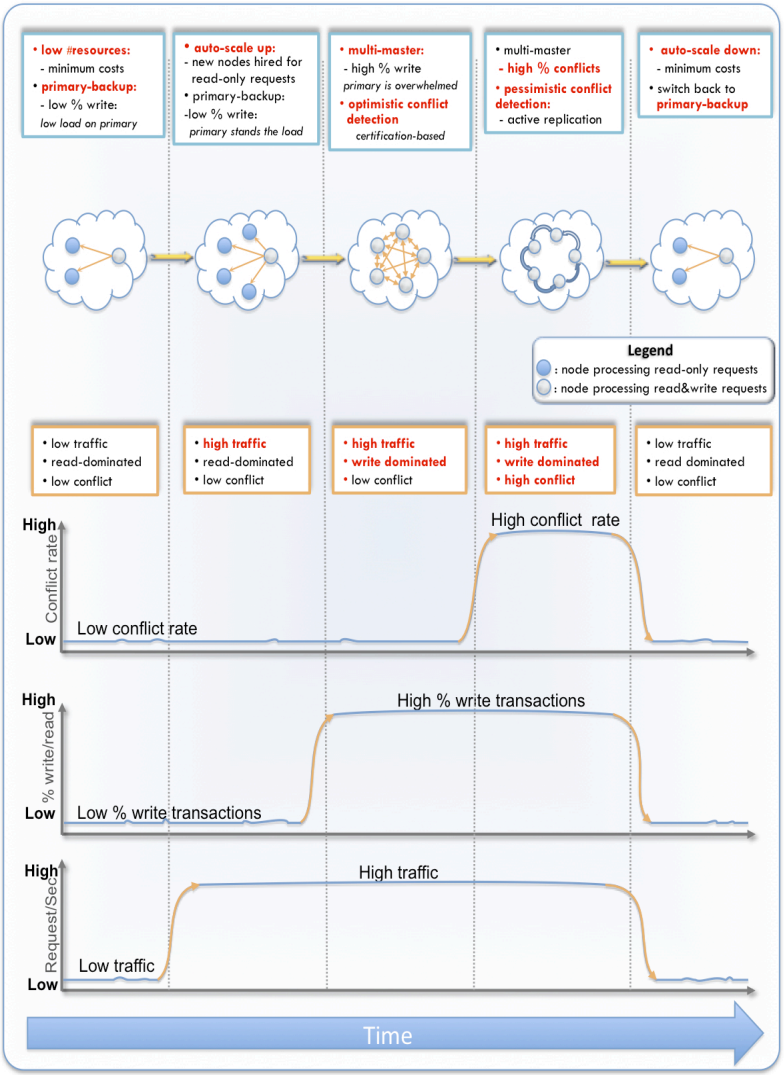
Five different workload scenarios are considered, each associated with distinct optimal management strategies minimising the number of

resources to be hired from the cloud (while, of course, ensuring the same, user-defined Service Level Agreements).

In the first scenario (from the left) the workload is read-dominated and both the incoming traffic and the transaction conflict probability are low. In these settings, the optimal strategy is to hire a low number of resources and to use a primary-backup replication scheme. As discussed in Section 2, this simple, centralised approach allows the minimising of the replica coordination overhead, providing optimal performances as long as the number of write transactions remains sufficiently low so to avoid overloading the primary.

As the load increases (second scenario from the left), leaving the conflict rate and the write/read ratio unaltered, the Cloud-TM will automatically scale up the number of nodes hired from the cloud infrastructure. The replication scheme does not vary with respect to the former scenario. In fact, being workload read-dominated, the primary is still able to quickly process the incoming write transactions.

Fig. 2.10. Example scenario of self-tuning replication in the Cloud-TM platform



In the following scenario (the third from the left), due to a surge in the relative number of write transactions, the primary gets overwhelmed by

the incoming write requests, becoming the system's bottleneck. The Cloud-TM will react by triggering a reconfiguration of the replication scheme, passing to employ a multi-master replication scheme which allows for an even distribution of the write requests among the nodes and, consequently, the achievement of a higher throughput. Since the transaction conflict rate remains low (because the write operations target low contended data regions, for example private user data), an optimistic conflict detection policy relying on a certification-based approach is adopted. In fact, by letting transactions optimistically access possibly stale data, this class of replication protocols allows for achieving maximum inter-replica parallelism, at least as long as the conflict rate remains sufficiently low (Ciciani et al. 1992).

In the fourth scenario, we consider precisely the increase of the transaction conflict rate. This type of workload may arise, for instance, in an e-auction application in proximity to the expiration of an auction for a highly contended item. As already hinted, in high-contention scenarios, optimistic conflict detection schemes suffer from a high abort rate and incur a significant performance degradation. The Cloud-TM middleware will detect such a suboptimal system configuration, and transparently switch to a more pessimistic conflict detection scheme, such as the active replication mechanism described in Section 2, which is known to perform better in high conflict scenarios by serialising transactions and reducing the transaction abort probability (Ciciani et al. 1992; Nicola and Jarke 2000).

Finally, to the extreme right, as the workload returns to exhibit its original characteristics (low load, write/read ratio and conflict rate), the Cloud-TM will accordingly release the additional nodes acquired from the cloud infrastructure and switch back to use a primary-backup scheme (that outperforms multi-master approaches in these workload conditions). This will avoid underutilizing the hired resources and minimise the operational costs sustained by the cloud user.

5. Current research directions explored in the Cloud-TM project

The design of self-tuning transactional replication mechanisms encompasses two major problems:

(1) determining the optimal replication strategy/operational mode on the basis of the current, and foreseen, workload and system characteristics
(2) allowing the efficient switching among multiple protocols/system configurations.

Concerning the first of the above two issues, we are faced with a complex multi-variable online optimisation problem that entails monitoring/tuning of a large number of system indicators/parameters.

The performance of a replicated transactional system is, in fact, not only determined by the current demand/utilisation of system resources, for example CPU, network bandwidth, memory. Workload characteristics (for example, transaction conflict probability), as well as scale (i.e. number of nodes) and topology (for example, single data centre vs multiple geographically distributed clusters) of the platform have a large impact on performance.

Further, self-tuning a replicated transactional system necessarily entails the automatic optimisation of at least some of its fundamental building blocks, such as the underlying Group Communication System (Miranda et al. 2001) and local concurrency control algorithm (Sonmez et al. 2009).

We are currently tackling this issue by using differentiated methodologies, including analytical performance modelling (Sanzo et al. 2011), machine learning techniques (using both online reinforcement learning techniques and off-line approaches, such as neural networks or decision trees [Couceiro et al. 2010]), and more recently control theoretic techniques (Goebel et al. 2009).

Some relevant results that we have recently achieved in these areas are related to the performance forecasting of popular transactional memory algorithms (Sanzo et al. 2011) (for example the commit-time locking algorithm used in TL2 [Dice et al. 2006]), of Atomic Broadcast protocols (Couceiro et al. 2009; Romano and Leonetti 2011), and of alternative certification replication schemes (Couceiro et al. 2011).

Let us now analyse the second issue, namely the need to design efficient mechanisms aimed at supporting the dynamic switching among different replication algorithms. Achieving this goal requires addressing problems of both a theoretical and practical nature.

On the practical side, it requires engineering a framework that permits efficient switching between heterogeneous (possibly) distributed state consistency mechanisms. On the theoretical side, it would be desirable to design ad hoc schemes to preserve correctness despite the simultaneous coexistence of incompatible contention management strategies.

In fact, unlike in conventional non-distributed transactional systems (Guerraoui et al. 2005), the Cloud-TM platform is composed by distributed nodes, which need to consistently cooperate according to a common replica coordination scheme. Generally speaking, this raises the problem of enforcing a distributed agreement among the platform nodes in case of alteration of the adopted replication protocol.

In practice, however, the complexity of this dynamic reconfiguration task depends on the nature of the replication protocols from/towards which the system transitions. In fact, Couceiro et al. (2011) show that by exploiting knowledge of the dynamics of the destination and target replication protocols, it is possible to architect the replication manager to simultaneously support multiple atomic broadcast based replication protocols. We are currently working on mechanisms allowing for seamless switching between single and multi-master strategies, avoiding any stall or unnecessary transaction abort during transition periods.

It is worth highlighting that the efficiency of the mechanisms employed to support the system's reconfiguration is of paramount importance. This is particularly true in case the self-tuning mechanism relies on online reinforcement learning techniques for identifying the reconfiguration policies to be used in the system.

These approaches are particularly attractive as they are based on lightweight algorithms, and since they spare the burden of developing accurate models of the system to be controlled. The latter is a complex, error-prone and time consuming task, which may even be unfeasible in virtualised environments (as typical of IaaS cloud infrastructures [Amazon 2010]), where little or no knowledge is available on the actual characteristics of the underlying physical infrastructure.

On the other hand, online learning approaches require (possibly periodical) exploratory phases, in which multiple reconfiguration strategies are tested in order to build (or enhance) the initial (or current) statistical knowledge of the system's behaviour (typically in terms of rewards associated with state/action pairs [Sutton and Barto 1998]). The practical viability of these approaches is therefore strictly dependent on the availability of mechanisms supporting efficient distributed reconfigurations.

6. Conclusions

Elastic scaling, one of the key drivers underlying the success of the cloud computing paradigm, makes the problem of self-tuning the data

consistency mechanisms a crucial issue in order to maximize efficiency of transactional data grids. This is one of the key goals being pursued in the Cloud-TM project, and on which this book chapter has focused.

Based on a critical analysis of the main existing transactional replication schemes, and via an experimental study relying on fully-fledged prototypes, we have highlighted a number of relevant trade-offs among the solutions proposed so far in literature. We have shed light on some of the key challenges that need to be addressed to achieve this ambitious goal, and overviewed some of the approaches that we are currently investigating to address these challenges.

References

Amazon. 2010. "Amazon elastic compute cloud amazon ec2". http://aws.amazon.com/ec2/ (accessed

Armbrust M., Fox A., Griffith R., Joseph A. D., Katz R. H., Konwinski A., Lee G., Patterson D. A., Rabkin A., Stoica I. and Zaharia M. "Above the Clouds: A Berkeley View of Cloud Computing." Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

Attiya, H., Gramoli, V. and Milani, A. "A provably starvation-free distributed directory protocol". In Proceedings of the 12th international conference on Stabilization, safety, and security of distributed systems (SSS'10). Springer-Verlag, Berlin, Heidelberg, 2010.

Bernstein, P. A., Hadzilacos, V. and Goodman, N. "Concurrency control and recovery in database systems." Boston: Addison-Wesley Publishing Co, 1986.

Bloom, B. H. "Space/Time Trade-Offs in Hash Coding with Allowable Errors." *Communications of the ACM* 13 (7): 422–426.

Bocchino, R., Adve, V. and Chamberlain, B. "Software transactional memory for large scale clusters". In Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming (PPoPP '08). ACM, New York, NY, USA, 2008.

Cachopo, J. and Rito-Silva, A. "Versioned Boxes as the Basis for Memory Transactions." *Science Computer Programming* 63(2) (2006):172–185.

Carvalho, N., Romano, P. and Rodrigues, L. "Asynchronous lease-based replication of software transactional memory". In volume 6452 of *Lecture Notes in Computer Science*, 376–396. Springer Berlin/Heidelberg, 2010.

Ciciani, B., Dias, D. and Yu, P. “Analysis of Concurrency-Coherency Control Protocols for Distributed Transaction Processing Systems with Regional Locality.” *Software Engineering, IEEE Transactions on* 18 (10) (1992): 899–914.

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, 2009.

Couceiro, M., Romano, P. and Rodrigues, L. “A machine learning approach to performance prediction of total order broadcast protocols”. In SASO’10: Proceedings of the 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Budapest, Hungary, 2010.

——. “Polycert: Polymorphic self-optimizing replication for in-memory transactional grids”. In Middleware ‘11: Proceedings of the ACM/IFIP/USENIX 12th Middleware Conference, Lisbon, Portugal, 2011.

Defago, X., Schiper, A., and Urban, P. “Total order broadcast and multicast algorithms: Taxonomy and survey.” *ACM Computing Surveys* 36 (4) (2004): 372–421.

Dice, D., Shalev, O., and Shavit, N. “Transactional locking ii.” In Distributed Computing, volume 4167 of *Lecture Notes in Computer Science*, edited by Dolev, S., 194–208. Springer Berlin / Heidelberg. 2006.

Franklin, M. J., Carey, M. J. and Livny, M. "Transactional Client-Server Cache Consistency: Alternatives and Performance." *ACM Transactions on Database Systems* 22 (3) (1997): 315–363.

Goebel, R., Sanfelice, R. and Teel, A. “Hybrid Dynamical Systems.” *Control Systems IEEE* 29 (2) (2009): 28–93.

Gray, J., Helland, P., O'Neil, P. and Shasha, D. “The dangers of replication and a solution”. In Proceedings of the ACM SIGMOD international conference on Management of data, J. Widom (Ed.). ACM, New York, NY, USA, 1996.

Guerraoui, R., Herlihy, M. and Pochon, B. “Polymorphic contention management.” In Distributed Computing, volume 3724 of *Lecture Notes in Computer Science*, edited by Fraigniaud, P., 303–323. Springer Berlin / Heidelberg, 2005.

Guerraoui, R. and Rodrigues, L. *Introduction to Reliable Distributed Programming*. New York: Springer, 2006.

Herlihy, M. and Sun, Y. “Distributed Transactional Memory for Metric-Space Networks.” *Distributed Computing* 20 (2007): 195–208.

Kallman, R., Kimura, H., Natkins, J., Pavlo, A., Rasin, A., Zdonik, S., Jones, E., Madden, S., Stonebraker, M., Zhang, Y., Hugg, J. and Abadi, D. “H-store: a high-performance, distributed main memory transaction processing system.” *Procedings of the VLDB Endowment* 1 (2) (2008): 1496–1499.

Kemme, B. and Alonso, G. “A Suite of Database Replication Protocols based on Group Communication Primitives.” In Proceedings of the The 18th International Conference on Distributed Computing Systems (ICDCS '98). IEEE Computer Society, Washington, DC, USA, 1998.

——. “Don’t Be Lazy, Be Consistent: Postgres-R, A New Way to Implement Database Replication.” In Proceedings of the 26th Very Large Data Base Conference, Cairo, Egypt. ACM, 2000.

Miranda, H., Pinto, A. and Rodrigues, L. “Appia, a Flexible Protocol Kernel Supporting Multiple Coordinated Channels.” In Procedings of ICDCS’01, Phoenix, Arizona. IEEE, 2001.

Nicola, M. and Jarke, M. “Performance Modeling of Distributed and Replicated Databases.” *IEEE Transactions on Knowledge and Data Engineering* 12 (4) (2000): 645–672.

Patino-Martínez, M., Jiménez-Peris, R., Kemme, B. and Alonso, G. “Scalable Replication in Database Clusters”. In Proceedings of the International Conference on Distributed Computing (DISC), 315– 329. New York: Springer, 2000.

Pedone, F., Guerraoui, R. and Schiper, A. “The Database State Machine Approach.” *Distributed and Parallel Databases* 14 (1) (2003): 71–98.

Romano, P. and Leonetti, M. “Self-tuning Batching in Total Order Broadcast Protocols via Analytical Modelling and Reinforcement Learning.” *ACM Performance Evaluation Review*, to be published 2011.

Sanzo, P. D., Ciciani, B., Palmieri, R., Quaglia, F. and Romano, P. “On the Analytical Modeling of Concurrency Control Algorithms for Software Transactional Memories: The Case of Commit-Time-Locking.” *Elsevier Performance Evaluation Journal*, to be published 2011.

Schneider, F. B. *Replication management using the state-machine approach*. ACM Press/Addison-Wesley Publishing Co., 1993.

Shavit, N. and Touitou, D. “Software transactional memory.” *Distributed Computing* 10 (1997): 99–116.

Sonmez, N., Harris, T., Cristal, A., Unsal, O. S. and Valero, M. “Taking the heat off transactions: Dynamic selection of pessimistic concurrency control”. In Proceedings of the 2009 IEEE International Symposium on

Parallel & Distributed Processing. Washington, DC, USA. IEEE Computer Society, 2009.

Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S., Hachem, N. and Helland, P. “The end of an architectural era: (it's time for a complete rewrite).” In Proceedings of the 33rd international conference on Very large data bases (VLDB '07). VLDB Endowment 1150–1160, 2007.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. Cambridge MA and London: MIT Press, 1998.