PolyCert: Polymorphic Self-Optimizing Replication for In-Memory Transactional Grids[⋆]

Maria Couceiro, Paolo Romano, and Luis Rodrigues

INESC-ID
Instituto Superior Tecnico, Technical University of Lisbon

maria.couceiro@ist.utl.pt, romano@inesc-id.pt, ler@ist.utl.pt

Abstract. In-memory NoSQL transactional data grids are emerging as an attractive alternative to conventional relational distributed databases. In these platforms, replication plays a role of paramount importance, as it represents the key mechanism to ensure data durability. In this work we focus on Atomic Broadcast (AB) based certification replication schemes, which have recently emerged as a much more scalable alternative to classical replication protocols based on active replication or atomic commit protocols. We first show that, among the existing AB-based certification protocols, no "one-fits-all" solution exists that achieves optimal performance in presence of heterogeneous workloads. Next, we present PolyCert, a polymorphic certification protocol that allows for the concurrent coexistence of different certification protocols, relying on machine-learning techniques to determine the optimal certification scheme on a per transaction basis. We design and evaluate two alternative oracles, based on parameter-free machine learning techniques that rely both on off-line and on-line training approaches. Our experimental results demonstrate the effectiveness of the proposed approach, highlighting that PolyCert is capable of achieving a performance extremely close to that of an optimal non-adaptive certification protocol in presence of non heterogeneous workloads, and significantly outperform any non-adaptive protocol when used with realistic, complex applications that generate heterogeneous workloads.

Keywords: Transactional systems, replication, autonomic computing, machine learning, atomic broadcast

1 Introduction

In-memory NoSQL transactional data grids are emerging as an attractive alternative to conventional relational distributed databases. By employing alternative data models, such as plain key/value stores, and relying on replication rather than on persistence to stable storage to ensure data durability, in-memory transactional data grids have shown

[⋆] This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds and the Aristos project (PTDC/EIA-EIA/102496/2008), and by the European Commission through the Cloud-TM project (FP7-257784).

to achieve higher performance, scalability, and elasticity when compared to classical SQL-based database management systems [32,28].

Replication clearly plays a role of paramount importance in these in-memory data platforms, as it represents the key mechanism to ensure data durability in face of unavoidable node failures. Unsurprisingly, replication algorithms employed in these in-memory platforms take inspiration from the vast literature on replication of transactional systems (traditionally, database systems [24,18,23], but also, more recently, transactional memory systems [8,7]).

Among the plethora of transactional replication mechanisms published in literature, over the last years, a wide body of research has highlighted that schemes based on Atomic Broadcast (AB) and certification [24,18,23] tend to outperform classic eager replication schemes based on distributed locking and atomic commit, which suffer from large communication overheads and are prone to thrashing due to distributed deadlocks [12]. Conversely, certification based schemes avoid any onerous replica coordination during the execution phase, running transactions locally in an optimistic fashion. The consistency of replicas (typically, 1-copy serializability [3]) is ensured at commit- time, via a distributed certification phase that uses a single AB to enforce agreement on a common transaction serialization order, avoiding distributed deadlocks, and providing non-blocking guarantees in the presence of failures. Furthermore, unlike active-replication approaches that require the full execution of update transactions at all replicas [30], only one replica executes an update transaction, whereas the remaining replicas are only required to validate the transaction and to apply the resulting updates. This allows to achieve high scalability levels even in the presence of write-dominated workloads, as long as the transaction conflict rate remains moderate [24].

In the design space of 1-copy serializable certification replication protocols, which represents the focus of this work, a decision that can have a dramatic impact on the actual efficiency and robustness of the system is related to how to address the trade-off between the size of the messages sent via the AB primitive and the number of communication steps required during the transaction commit phase. Depending on how this trade-off is addressed, existing certification-based replication algorithms can be classified into three main categories:

- Solutions that disseminate the whole transaction's read-set to all replicas, called *Non-voting* schemes, allow each replica to certify transactions locally, by sending both the read-set and write-set via an AB primitive. This makes these protocols optimal in terms of communication steps, but also makes them prone to generate very large messages and to overload the Group Communication System.
- *Voting* schemes, which avoid broadcasting the read-set of transactions by sending (via AB) only the write-set, thus drastically reducing the network bandwidth consumption. On the other hand, they incur into the costs of an additional coordination phase along the critical path of the transaction commit , which can hamper significantly performance.
- Approaches relying on the space efficient encoding of Bloom Filters [4] to implement a variant of the non-voting certification mechanism, called *Bloom Filter Certification* (BFC) [8] . Unlike voting mechanisms, BFC determines the outcome of transactions using a single AB, generating smaller messages when compared to

non-voting protocols. The probabilistic nature of the Bloom filter encoding, however, induces false positives in the certification phase, increasing the transaction abort rate.

The above protocols are designed to ensure optimal performance in different workload scenarios and, as we will show in the following, they can exhibit up to 10x differences in terms of maximum throughput. Our goal is to alleviate the developers/administrators from the hard and time-consuming task of profiling the application and selecting the most suitable replication protocol for each deployment. Furthermore, a static configuration may lead to largely suboptimal configurations in presence of heterogeneous workloads. In these contexts, the employment of a single, statically chosen, replication mechanism, optimized for a specific workload type, will lead to suboptimal performance when processing the transactions that have different characteristics.

The solution presented in this work, which we named *Polymorphic Self-Optimizing Certification* (PolyCert), supports the simultaneous use of the three aforementioned classes of protocols, and relies on machine-learning techniques to determine, on a per transaction basis, the certification strategy to be adopted. PolyCert relies on a modular design, which encapsulates the logic associated with the on-line choice of the replication strategy into a generic oracle. We design and evaluate two alternative mechanisms to implement this oracle, based on two different parameter-free statistical learning techniques.

- An off-line technique based on regression decision trees [26], that requires a preliminary, computational intensive, feature selection and training phase, but that was shown (in our previous work [9]) to achieve high accuracy in forecasting the performance of Atomic Broadcast algorithms in presence of heterogeneous workloads.
- An on-line reinforcement learning technique, that uses an innovative, parameter-free variant of a very lightweight, but theoretically optimal solution [2] to face the exploration versus exploitation dilemma, i.e. the search for a balance between exploring the environment to find profitable actions while taking the empirically best action as often as possible.

Via an extensive experimental evaluation, based on a fully fledged system prototype and a range of heterogeneous benchmarks, we assess the effectiveness of the proposed approaches in terms of performance benefits and learning time. We show that PolyCert can achieve a significant speed-up with respect to static solutions and enhance the robustness of the system to unexpected fluctuations of the workload.

The remainder of the paper is structured as follows. Section 2 reports the results of a performance evaluation study highlighting the relevance of the addressed problem. The system architecture is presented in Section 3. Section 4 describes the functioning of PolyCert and the mechanisms employed to determine at run-time which certification strategy to use. The results of the experimental evaluation study are reported in Section 5. Related work is analysed in Section 6. Section 7 concludes the paper.

2 Motivations

As already mentioned in the Introduction section, existing certification-based solutions can be classified into three main categories:

- **Non-Voting Certification (NVC):** These solutions [24,7,23] disseminate the whole read-set and write-set using the AB service, allowing every replica to determine, upon delivery of the corresponding message, the outcome (commit/abort) of the transaction, by running the certification procedure locally. These schemes are optimal in terms of communication steps, delivering excellent performance when used in workloads characterized by small transaction read-sets. On the other hand, they exhibit very poor performance in presence of transactions reading a significant number of data items. Even worse, in these scenarios, the large network traffic generated by this protocol can saturate and disrupt the proper functioning of the Group Communication Service, leading to network partitions and false failure suspicions.
- **Voting Certification (VC):** These solutions [17] disseminate exclusively via AB the transaction write-set, thus avoiding the issues incurred in by Non-voting schemes with large transaction read-sets. On the down side, the transaction can only be certified at the site in which it was originated. This implies the need for an additional communication phase, executed using a Uniform Reliable Broadcast (URB) [14] (a lighter communication primitive when compared to AB), which is triggered by the replica where the transaction was originated in order to inform the remaining replicas of the final outcome of the transaction. This extra communication phase, which requires at least two communication steps, has a negative impact on the latency of the commit phase, which represents by far the dominating cost for small transactions. By introducing additional latency in the critical path of the commit phase, which needs to be run sequentially for conflicting transactions, these schemes can adversely affect the maximum throughput achievable by the platform [29].
- **Bloom Filter Certication (BFC):** An alternative approach, denoted as Bloom Filter Certification (BFC) [8], consists in encoding the read-set of the transaction in a Bloom filter [4], a space-efficient data structure that allows compressing 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 significantly the size of the messages exchanged via the AB service when compared to non-voting schemes. On the down side, BFC can suffer from false positives due to the probabilistic nature of Bloom filter-based encoding, which ultimately leads to an additional rate of aborted transactions.

From the above discussion, 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. Unfortunately, realistic transactional applications can exhibit very heterogeneous workloads encompassing read-sets whose sizes range from less than ten to hundreds of thousands of objects. We have experimentally observed this phenomena, as illustrated in Figure 1, which depicts the distribution of the read-set size for a widely used benchmarking application for in-memory transactional systems (in particular Transactional Memories), namely the STMBench7 [13] benchmark.

In Figure 2 we show the results of a sensitivity analysis aimed at assessing the actual impact of the read-set size distribution on the performance of the three certifi-

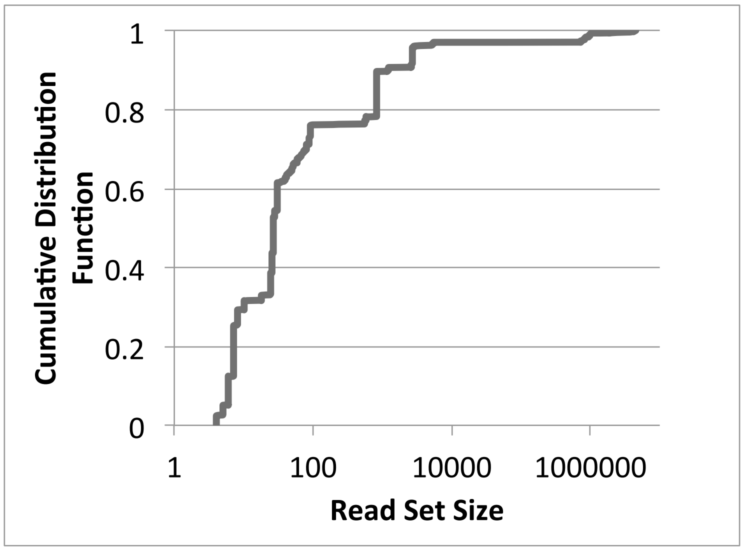

Fig. 1. Distribution of transaction read-set size in the STMBench7 Benchmark.

cation schemes described above. The results were obtained using a simple synthetic benchmark adapted from the Bank Benchmark originally used in [8]. This benchmark simulates the concurrent transfer of funds from different bank accounts (modelled as a simple array of doubles), and was altered to vary the number of items read within a transaction in the range [1,100'000]. Further, to focus only on the effects due to variations of the read-set size, which represents the goal of this sensitivity analysis, we configured the benchmark 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 [8]). These results were obtained running 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 (which represents the reference experimental platform used in the remainder of the paper). 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 [6], used to manage local concurrency, and ii) a replicated key/value store, used to maintain associations between unique object identifiers and object instances. Further details on the system architecture will be provided in Section 3.

Our experimental results highlight that no-one-fits-all solution exists that maximizes the throughput across all the considered workloads. On the contrary, NVC provides the best performance in the scenario with small read-sets, BFC in the scenario with 1000 items in the read-set, and VC is by far the best performing protocol 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 1000) to 10x (VC vs NVC, read-set size equal to 100'000).

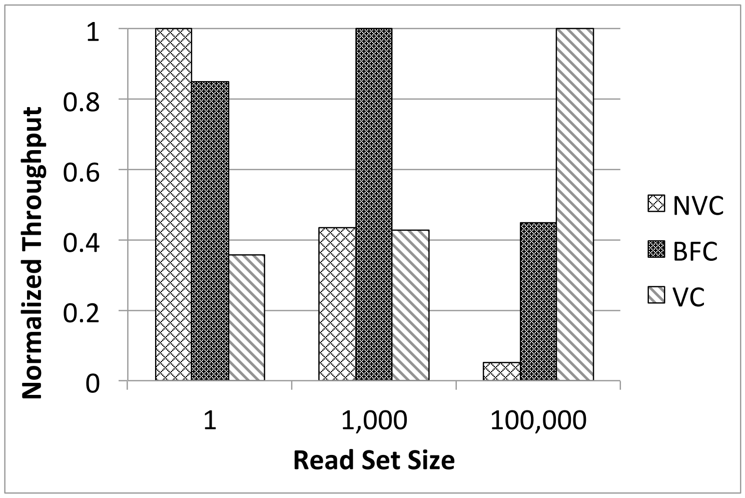

Fig. 2. Throughput of three certification strategies with different read-set sizes.

3 System Architecture

The system architecture is depicted in the diagram shown in Figure 3. At the topmost layer, it exposes the API of an object-oriented STM, which is however fully replicated across a number of distributed nodes. The API provided to applications is a transparent extension of JVSTM's API, a state-of-the-art STM relying on an efficient Multi Version Concurrency Control (MVCC) algorithm [3]; a strong advantage of JVSTM is that read-only transactions are never required to block. The JVSTM programming paradigm requires that the programmer encapsulates any shared mutable state within VBoxes, which are then managed by JVSTM's MVCC to ensure transactional atomicity and isolation. This allows separating the transactional and non-transactional state of the application, ensuring strong atomicity [19] at no additional costs. We modularly extended JVSTM by augmenting it with what we have named a Polymorphic Replication Manager (PRM).

The PRM is in charge of triggering the execution of a certification protocol for each of the locally executed transactions, and to participate in the certification of transactions that have been executed at remote nodes. A unique feature of PRM, with regard to existing replication managers, is that it is able to determine, for each locally executing transaction, which certification algorithm is more appropriate, given the characterization of the transaction. The logic for determining the certification scheme is encapsulated by the abstraction of a Replication Protocol Selector Oracle (RPSO), whose interface exports two main functionalities:

– Given a local transaction, it selects the most appropriate certification protocol to be executed by the PRM. In Section 4.1 we will present and evaluate two performance forecasting methods which are based on alternative machine learning techniques.

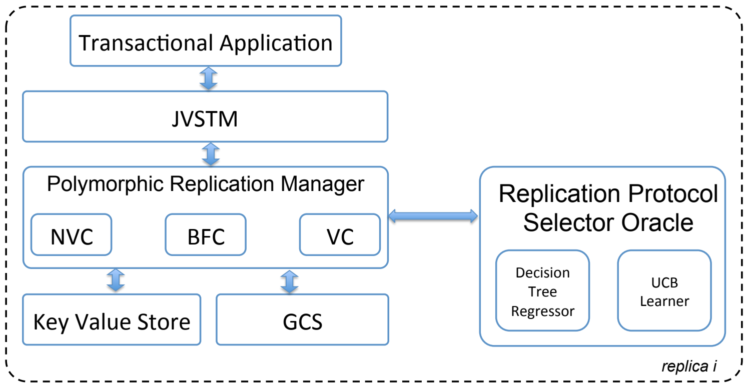
Transactional Application
JVSTM
Polymorphic Replication Manager
NVC
BFC
VC
Key Value Store
GCS
Replication Protocol Selector Oracle
Decision Tree Regressor
UCB Learner
replica i

Fig. 3. Architectural Overview (Single Node Perspective).

– Collecting historical data about the execution of past transaction, to improve the selection process. For this, it exports an interface that allows the PRM to register the following information: i) the commit time experienced by a transaction, and ii) the certification protocol that it used. This allows the RPSO to gather, store and analyse (either on-line or off-line) statistical data on the distribution of the commit time of transactions.

The PRM also interacts with two other components, the Group Communication System (GCS) and a local Key Value Store (KVS).

The GCS, Appia [21] in our implementation, provides a number of communication abstractions required by the PRM: view synchronous membership, AB and URB [14].

Finally, the Key Value Store is a weak hash map, used to maintain the mapping between application level transactified objects (namely, containing JVSTM VBoxes) and replica-wide unique object identifiers, which are generated automatically by our framework upon creation of a new transactional object. More in detail, an entry of the key/value store contains the unique object identifier, as its key, and a *weak reference* to the local transactional object as its value. This information is used by the PRM, upon reception of a commit request for a remote transaction T, to retrieve in the local JVSTM instance the objects that were read/updated by T during its remote execution. The usage of a weak hash map ensures that the Java garbage collector is not prevented from discarding the object referenced by a hash map entry whenever all application level references to the object have been removed, thus avoiding any interference with the local JVM garbage collection mechanism.

4 The PolyCert replication protocol

PolyCert is designed to allow the simultaneous use of all the three AB-based certification protocols introduced before, namely NVC, BFC and VC. Specifically, at commit time (when the size of the transaction read-set and write-set is already known), the PRM invokes the RPSO to determine which certification protocol to use in order to finalize the transaction's execution. This clearly implies that concurrent transactions may use different protocols, which need to coexist without endangering the correctness of the system. The three protocols lend themselves quite naturally to coexist, given that all of them rely on a first common phase during which they establish, via the AB primitive, the global serialization order for a committing transaction (even though each protocol piggybacks different information). Naturally, certification messages need to be tagged with a label that specifies which of the protocols is being used for each transaction.

Upon delivery of an AB message, the PRM performs the following steps:

– The message is inserted in a queue containing the transactions to be certified.
– If the transaction is being certified using NVC or BFC, no further processing is done until the message reaches the head of the queue.
– If the transaction is being certified using VC, and the node was the originator of the transaction, the following procedure is executed immediately. Upon the enqueuing of a transaction, say T, if and only if T does not conflict with any of the transactions already present in the certification queue (i.e. the read-set of T does not intersect with the write-sets of the transactions already present in queue), T is immediately validated, by verifying whether the snapshot that it read is still fresh, the URB conveying the decision outcome is triggered. This optimization, originally proposed in [29], allows to overlap the URB dissemination phases of (non-conflicting) transactions with the time spent by transactions in the certification queue, reducing the risk of convoy effects which are otherwise known (and confirmed by our experience) to significantly hamper performance of VC schemes.

Subsequently, messages are removed from the head of the queue one by one and the corresponding transactions validated in a sequential manner. More specifically:

– If the message that is extracted from the head of the queue corresponds to a transaction that is being certified using NVC or BFC, each node applies locally the corresponding certification algorithm. Essentially, it verifies if the read-set accessed by the transaction is still valid, and commits or aborts the transaction accordingly.
– If the message that is extracted from the head of the queue corresponds to a transaction that is being certified using VC, the node checks if a vote has already been received or not. If a vote has been received and the decision was to commit the transaction, the write set is applied. Otherwise, if the vote has been received and the decision was to abort the transaction, the write-set is discarded (actually, the transaction can be immediately removed from the queue as soon as an abort vote is received). On the other hand, if a vote has not been received and the transaction is remote, the certification is suspended until the corresponding vote is received. Finally, if a vote has not been received and the transaction is local, the node validates the transaction as described above, and issues the vote (this corresponds to the scenarios where the optimization described before cannot be applied).

As a final remark, note that, since all replicas deliver the certification messages in the same order due to the use of the AB primitive, and decide deterministically which certification protocol to use, consistency is guaranteed even in presence of concurrent transactions being processed using different certification schemes.

4.1 Replication Protocol Selection Oracle

As already mentioned, the Replication Protocol Selector Oracle abstraction (RPSO) is a convenient form of encapsulating different performance forecasting techniques. In this paper, we present two implementations of the RPSO, using two different machine learning techniques, namely a regressor based on decision trees [26], and a reinforcement learning technique, namely UCB [2], as described below.

Oracle based on regressor decision trees In order to forecast the time necessary for committing a transaction with each of the three considered certification strategies, we start by forecasting the size, m_p, of the AB message that would be generated by each of the certification protocols $p \in \{NVC, BFC, VC\}$. This corresponds to the number of bytes required to transmit the transaction read-set and write-set with NVC, the transaction write-set with VC, and the write-set and the Bloom filter based encoding of the transaction read-set with BFC.

Next, we forecast the time for marshalling and validating a transaction with each of the considered certification schemes. To this end, we maintain, for each certification strategy, a moving average of the average marshalling time per byte, denoted as T_p^m, and of the validation time, denoted as T_p^v, for all $p \in \{NVC, BFC, VC\}$. Further, for BFC, we maintain moving averages of the time required to build a Bloom filter that encodes the read-set (normalized by the read-set's size), denoted as T^{BF}. Finally, for VC, we maintain also the moving averages of the self-delivery latency of the URB convoying the vote from the transaction's initiator, denoted as T^{URB}. Note that the choice of measuring self-delivery latencies allows us to avoid distributed clock-synchronization mechanisms, which in our preliminary experiments revealed not to be sufficiently accurate for our purposes.

Using the metrics above, the commit latency T_p for a transaction using certification protocol p is then forecast as follows:

$$T_{NVC} = T_{NVC}^m \cdot m_{NVC} + T_{NVC}^v + T^{AB}(m_{NVC}) \quad (1)$$

$$T_{BFC} = T_{NVC}^m \cdot m_{BFC} + T_{VC}^v + T^{BFC} \cdot rsSize + T^{AB}(m_{BFC}) \quad (2)$$

$$T_{VC} = T_{VC}^m \cdot m_{VC} + T_{VC}^v + T^{AB}(m_{VC}) + T^{URB} \quad (3)$$

where $T^{AB}(m)$ is the forecast latency for self-delivering a message of size m using the AB primitive. To this end, we exploit our recent results in [9], where we presented and evaluated a series of (off-line) machine learning techniques to forecast AB's performance, including neural-networks [16] and support vector machines [31]. In the light of the results achieved in [9], we integrated in our system a regression technique relying on the Cubist© [25] decision-tree regression algorithm, which proved to be the most accurate and robust predictor among those evaluated.

Analogously to classic decision tree based classifiers, such as C4.5 and ID3 [26], Cubist© builds decision trees choosing the branching attribute such that the resulting split maximizes the normalized information gain (namely the difference in entropy). However, unlike C4.5 and ID3, which contain an element in a finite discrete domain (i.e. the predicted class) as leafs of the decision tree, Cubist© places a multivariate linear

Metric	**Description**
freeMem	Free memory in the Java Virtual Machine
tLGC	The time since the last garbage collection
pLGC	% of time since the last GC cycle w.r.t. the time between the last 2 GC cycles
undelivMsgs	#TO Broadcast msgs and not yet self-delivered
bytesUp$_x$	#Bytes received over a x msec. time window
bytesDown$_x$	#Bytes sent over a x msec. time window
TOBUp$_x$	#TOB deliver events over a x msec. time window
TOBDown$_x$	#TOB broadcast events over a x msec. time window
totCPU$_x$	% total CPU utilization over a x msec. time window
esCPU$_x$	% CPU utilization by ES thread over a x msec. time window
TCPqueue	Outgoing messages queued at the Transport Layer

Table 1. List of metrics (features) collected by the Monitoring Layer.

model at each leaf, which we use to predict the AB self-delivery latency (expressed in microseconds).

In order to generate the training data for the decision tree regressor we ran the synthetic benchmark described in Section 2, for each of the three considered certification protocols, varying every 3 minutes the read-set size of the generated transactions in the set $\{$10,100, 1'000, 100'000$\}$. Overall the training data set gathered by each replica is constituted, on average, by approximatively 25'000 samples, reporting the self-delivery latency for each AB message along with the message size, and a total of 53 different metrics (i.e. context information), also referred to as features, including averages on multiple time scales and time series of a plethora of metrics (listed in Table 1) concerning the utilization of various system resources (CPU, RAM and Network). The choice of this synthetic benchmark to generate the training data set has the following rationale: since this benchmark generates transactions with very heterogeneous read-set sizes, it allows gathering a good a-priori knowledge on the performance of a wide range of possible workload scenarios that the system may face when running more complex, realistic applications.

To minimize the effects of overfitting, which are likely to occur given the high dimensionality of the feature space, we run a greedy feature selection algorithm (Forward Selection [15]) aimed at discarding loosely correlated features and boosting the predictor's accuracy. Feature selection is by far the most time consuming phase of the off-line training, taking on average 45 minutes (per replica) when run on a PC equipped with Intel Core 2 CPU with a 2.2GHz clock-rate and 2GB of RAM. On the other hand, feature selection allows to achieve a significant improvement in the accuracy of the predictions, as highlighted by the results shown in Table 2, which report the correlation factor and mean absolute error using 10-fold cross-validation before and after performing feature selection.

Oracle based on the UCB online learner The second implementation of the Oracle Layer employs an on-line learning technique. Therefore, it does not require an a-priori

Metric	**Before FS**	**After FS**
Relative Absolute Error	0.81	0.30
Correlation Coefficient	0.17	0.76

Table 2. Prediction accuracy of the decision tree regressor before and after feature selection.

computational intensive off-line training. Instead, it relies on a lightweight reinforcement learning (RL) technique that updates the knowledge of the oracle as the system is running.

This oracle relies on a customized, self-tuning version of a state of the art RL algorithm, called UCB (Upper Confidence Bounds), which solves (in a theoretically optimal manner) a classical on-line learning problem, known in literature as the *multi-armed bandit* [27]. 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. Formally, each arm i of the bandit, for $0 \leq i \leq k$, is associated with a sequence of random variables $X_{i,n}$ representing the reward of the arm i, where n is the number of times the lever has been used. The goal of the agent is to learn which arm i maximizes the criterion:

$$\mu_i = \sum_{n=1}^{\infty} \frac{1}{n} X_{i,n}$$

that is, achieves maximum average reward. To this purpose, the learning algorithm needs to try different arms in order to estimate their average reward. On the other hand, each suboptimal choice of an arm i costs, on average, $\mu^* - \mu_i$, where μ^* is the average obtained by the optimal lever. Several algorithms have been studied that minimize the *regret*, defined as

$$\mu^* n - \mu_i \sum_{i=1}^{K} E[T_i(n)]$$

where $T_i(n)$ is the number of times arm i has been chosen. Building on the idea of *confidence bounds*, the UCB algorithm creates an overestimation of the reward of each possible decision, and lowers it as more samples are drawn. Implementing the principle of *optimism in the face of uncertainty*, the algorithm picks the option with the highest current bound. Interestingly, this allows UCB to achieve a logarithmic bound on the regret value not only asymptotically, but also for any finite sequence of trials [2].

More in detail, UCB assumes that rewards are distributed in the [0,1] interval, and associates each arm with a value:

$$\overline{\mu_i} = \overline{x_i} + \sqrt{2 \frac{\log n}{n_i}} \tag{4}$$

where $\overline{x_i}$ is the current estimated reward for arm i, n is the number of the current trial, n_i is the number of times the level i has been tried. The right-hand part of the sum in Eq. 4 is an *upper confidence bound* that decreases as more information on each option is

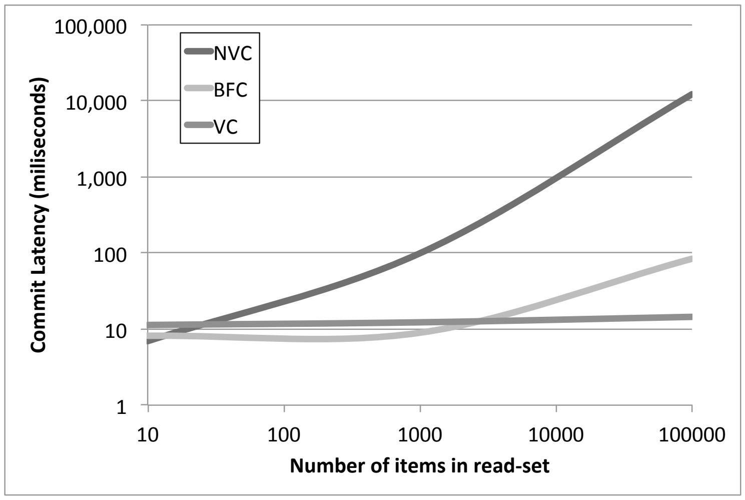

Fig. 4. Commit latency as a function of the read-set size.

acquired. By choosing, at any time, the option with maximum $\overline{\mu_i}$, the algorithm searches for the option with the highest reward, while minimizing the *regret* along the way.

In order to apply the UCB technique to our problem, we had two face two main issues, which we discuss in the following paragraphs.

State space discretization. As we have illustrated in Section 2, the performance of certification depends on the workload characterization. Thus, using a single UCB instance, having as arms the three alternative protocols for all possible scenarios is clearly not a viable solution. This observation raises the problem of discretizing the workload state space into a set of distinct, representative, classes of workload scenarios. This allows, in fact, to associate a different instance of a UCB learner with each discretized interval of the workload's parameter space, and to train each instance to choose among the 3 considered protocols under specific workload conditions.

The discretization process involves a delicate trade-off: a finer (i.e. denser) discretization can lead, eventually, to more accurate predictions across the entire state space, but requires the training of a larger number of UCB instances, which can lead to a considerable increase of the learning time. In order to determine an appropriate discretization strategy, we analysed the average commit latency of each of the three protocols as a function of the read-set size using the synthetic benchmark introduced in Section 2. The results, reported in the log-log plot of Figure 4, highlight that, for NVC and BFC, the read-set size and commit latency exhibit a heavy-tail relationship. At the light of this observation, we opted to use the read-set size as the reference variable to discriminate different workload situations,and we discretized it using exponentially increasing intervals, where each sampling interval is defined by the range $[10^i, 10^i + 1]$ with $i \in \{1 \ldots 6\}$). This choice allowed us to partition the state space into a small number of intervals, thus reducing learning time, while associating each discretized interval with fluctuations of approximately the same relative amplitude in the commit latency, even for the case of the NVC, whose commit latency is the most sensible to variations of the read-set size.

Definition of the reward function. UCB is based on the assumption that rewards are distributed in the [0,1] interval, whereas, as we have seen in Figure 4, the commit latencies are distributed over a very large domain. This required defining a mapping function, denoted as $R(t)$, taking as input a commit latency, t, and outputting a value (the reward) distributed in the [0,1] interval. In order to preserve the relative distance among samples before and after applying the mapping function we employed the following linear transformation:

$$R(t) = \frac{maxLatency - min\{maxLatency, t\}}{maxLatency}$$

which relies on the parameter $maxLatency$, defining a threshold for the commit latency, above which the reward is mapped to the value 0. Based on our preliminary experiments, we observed that the correct definition of the $maxLatency$ parameter value has a fundamental impact on the effectiveness of UCB: excessively low or high values would in fact lead to saturating the reward function, preventing UCB to distinguish sensibly the performance of the various protocols. Also, the manual tuning of this parameter is an extremely time-consuming task, given that the setting of $maxLatency$ was found to depend strongly on the characteristics of the user level application. For instance, we noted that, when testing this approach with the STMBench7 benchmark, we had to increase the value of $maxLatency$ by a factor approximately 27x larger than when using the synthetic benchmark described in Section 2.

This led us to define a self-tuning mechanism to define the value of the $maxLatency$ parameter. This mechanism is based on the observation that the (average) commit latency when using VC is i) largely unaffected by the read-set size (given that it does not disseminate the read-set), and ii) lower than that of *both* NVC and BFC for sizes of the read-set larger than some threshold (this threshold being unknown and dependant on the application and deployment scenario). In other words, VC's commit latency represents a consistent upper bound for NVC's and BFC's commit latencies below a given read-set's size threshold, in which the two protocols typically exhibit alternate performances. On the other hand, it represents a lower bound for NVC's and BFC's commit latency for high read-set's size, a scenario in which it is actually unnecessary to be able to accurately predict their performance, given that VC outperforms them significantly.

This makes the VC's average commit latency, denoted as T_{VC}, a good reference point for UCB's $maxLatency$ parameter value. This insight led us to define the following rule:

$$maxLatency = \overline{T_{VC}} \cdot (1 + \sigma(T_{VC}))$$

where $\sigma(T_{VC})$ denotes the standard deviation (more precisely the squared root of the sampling variance) of T_{VC}. In order to instantiate this formula, upon boot-strapping of the system, we execute transactions using the VC scheme until the following stopping condition is reached:

$$\sigma(T_{VC}) < 2 \cdot \overline{T_{VC}}$$

which in our experiments typically implied a few tens of transactions (and that was however upper bounded to 100 transactions to ensure robustness in the presence of highly disperse sampling data). To minimize the impact of this (typically quite short) bootstrapping phase on the learning time, we provide the observed sampling data to the corresponding UCB's instances also during this phase (in which UCB's instances are not being queried to choose the replication protocol), thus allowing them to gather statistical information concerning the reward of the arm associated with the VC protocol.

A further optimization that we designed in order to minimize learning time is to have the replicas periodically exchange and merge the locally gathered statistical information concerning the reward distributions of UCB's arms. This allows the replicas to mutually benefit from the statistical knowledge that they have gathered so far, narrowing the upper confidence bounds of the UCB's instances and accelerating their convergence. To minimize the overhead, we piggyback periodically (e.g. each 10 seconds in our experiments) the state of the 6 UCB's instances maintained at each replica (encoded by the tuple $< \overline{x_i}, n_i, n >$ for each of its three arms $i \in \{NVC, BFC, VC\}$, and globally accounting to around 100 bytes) to the AB messages generated by the PRM. As soon as updated statistical information from a different replica is received, the information concerning the local UCB instances is updated by setting, for each arm i:

- the value of $\overline{x_i}$ to the average of the local and remote values of $\overline{x_i}$, weighted proportionally to the number of times i was played locally and remotely, namely:

  $$\overline{x_i} = w_i^{loc}\overline{x_i} + w_i^{rem}\overline{x_i^{rem}}, \text{ where } w_i^{loc} = \frac{n_i}{n_i+n_i^{rem}} \text{ and } w_i^{rem} = 1 - w_i^{loc}$$

- the value of n_i and n to the sum of their, respectively, local and remote values, namely $n_i = n_i + n_i^{rem}$ and $n = n + n^{rem}$

5 Experimental Evaluation

In this section we report the results of an experimental study aimed at assessing the performance gains achievable by PolyCert, and the adequacy of the proposed machine-learning based self-optimizing mechanisms.

We start by considering the synthetic benchmark already used in Section 2 that, thanks to its simplicity and predictability, allows us to analyse the performance of PolyCert in precisely identifiable workload scenarios. We then evaluate a widely used benchmark for Transactional Memories, namely STMBench7, already mentioned in Section 2. STMBench7 is a complex benchmark that features a number of operations with different levels of complexity over an object-graph with millions of objects, generating a very intense and heterogeneous workload for the GCS. All the throughput results reported in the following were obtained averaging over a number of runs sufficient to ensure that the width of the 90% confidence intervals for the throughput was less than 10% of the corresponding average value.

The bar plot in Figure 5 reports the normalized throughput (with respect to the optimal non-adaptive workload) for each of the workloads generated by the Bank benchmark, including the versions of PolyCert when employing the oracles based on regressor

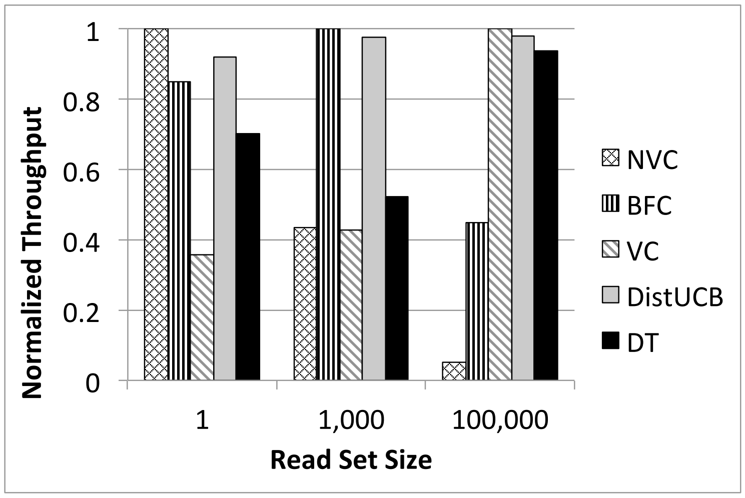

Fig. 5. Normalized throughput of the adaptive and non-adaptive protocols (Bank benchmark).

decision trees and UCB (respectively dubbed as DT and DistUCB in the plot). These experiments were run by switching the workload every three minutes, thus the reported performance incorporates also data gathered during the initial phases during which we bootstrap the statistical information of UCB.

Our experimental data shows that the on-line learning oracle using UCB (with the optimization for periodically exchanging statistical information among replicas enabled) achieves a performance very close to the corresponding optimal protocol for each scenario, namely on average around 5% less than the optimal solution and in the worst case, the scenario where the transaction's read-set size is set equal to 1, less than 10% from the optimum. In this scenario, UCB alternates between BFC and NVC, whose performances are quite close (differing by around 15%); in several runs some replicas eventually converged towards the choice of BFC. In all the remaining scenarios, after a short bootstrapping phase, the replicas converged consistently towards the choice of the optimal certification protocols, which explains why they achieved performance almost indistinguishable from those of an optimally tuned non-adaptive protocol.

On the other hand, the performance achieved by the oracle based on regressor decision trees was significantly worse. When using DT, the performance of PolyCert was approximately 25% worse than that of the corresponding optimal non-adaptive scheme (across the three workloads). Note that, DT was still able to outperform the second best non-adaptive protocol (but not the optimal choice). A main source of inefficiency in the implementation of the DT oracle is the following: it relies on the Java Native Interface (JNI) to query the decision tree-based model generated by Cubist, implemented in C. The overheads due to JNI are negligible in the scenario with read-set size equal to 100K, whose transactions have a local execution time in the order of a few tens of milliseconds. On the other hand, JNI's overheads have a negative impact on performance in the scenarios with smaller read-set sizes, in which transactions have a local execution time

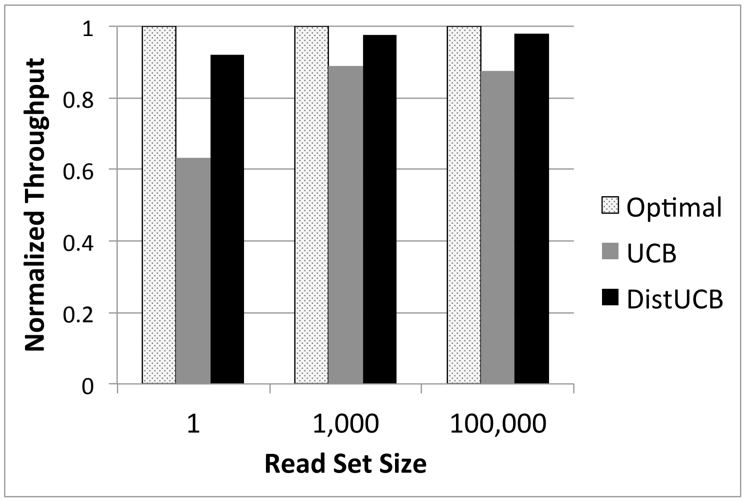

Fig. 6. Normalized throughput of UCB and DistUCB over a three minute run (Bank benchmark).

on the order of just a few tens of microseconds. The performance of DT is lower in the scenario with a read-set size of 1000, as in this case the DT oracle had a lower accuracy in forecasting the AB self-delivery time, and erroneously biased its decisions towards the voting protocol (which is chosen in approximately 30% of the cases on average).

In Figure 6 we contrast the performance of the UCB oracle (again in terms of normalized throughput vs the optimal non-adaptive protocol), over a three minute run, with and without enabling the optimization of exchanging periodically (each 10 seconds) statistical information among replicas to improve learning. The data clearly shows the effectiveness of this optimization, with speed-ups larger than 25% due to the fastest convergence towards the optimal non-adaptive solution. Figure 7 provides more detailed insights on the speed of convergence of UCB and DistUCB versus the optimal solution, reporting the average throughput over 10 seconds time windows, achieved by the two protocols. The plots clearly highlight the positive effects, in terms of learning time reduction, due to the exchange of statistical information occurring, in particular, at the time instants 10, 20 and 30 (seconds), that nearly halves the time required to converge to the optimal choice.

Finally, we assess the performance of PolyCert with STMBench7, plotting the corresponding results in Figure 8. The benchmark was configured to use the write-dominated workload with long traversals, which generates approximately 90% of update transactions, thus allowing us to focus on the performance of the transactions that require the activation of a commit-time certification phase. As shown in Figure 1, around 5% of transactions (namely the so-called *long traversal* transactions) in this benchmark have read-set sizes larger than 500K items. As a consequence, when using either NVC or BFC, this benchmark generates a very high traffic volume that, in all our runs, eventually determined the saturation and the collapse of the GCS. This is the reason why in Figure 8 we only report the throughput of VC, DT, UCB and DistUCB (nor-

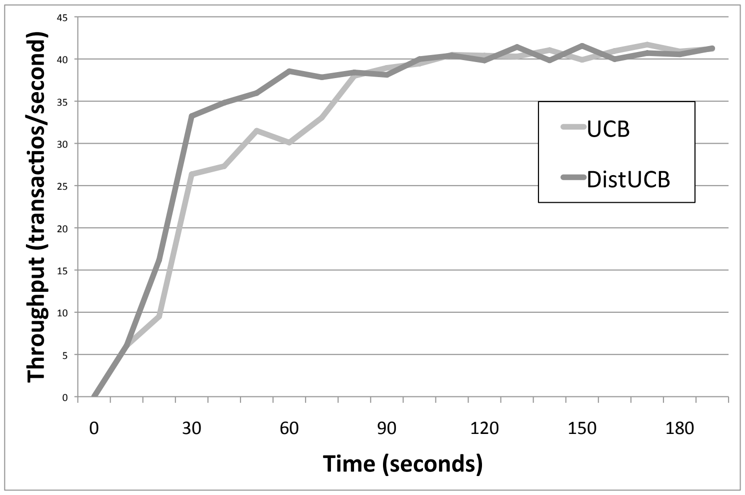

Fig. 7. Evolution of throughput over time with UCB and DistUCB (Bank benchmark - 100K read-set size scenario).

malized with respect to the throughput of the optimal non-adaptive protocol, namely VC). In this scenario, the adaptive protocols clearly outperform the non-adaptive VC scheme, thanks to their ability to use the more efficient NVC and BFC protocols to handle transactions with smaller read-set's size. The speed-up of PolyCert when using the three alternative oracles ranges from 25% to 35%, with the best performance also in this case achieved by DistUCB.

Overall, our experimental data demonstrated the effectiveness and viability of the proposed self-tuning polymorphic replication technique. The reported results highlight in particular the efficiency of the DistUCB oracle, which, not needing any time-consuming off-line training phases, and being totally parameter-free, results as extremely convenient for deployment in real-life practical scenarios. Interestingly, PolyCert does not only provide benefits in terms of performance, but also in terms of robustness, avoiding to saturate the GCS in presence of transactions with extremely large read-sets, a main source of instability for BFC and, in particular, NVC.

6 Related Work

Our work is clearly related to the vast literature on replication of transactional systems, and in particular to the more recent works relying on AB to achieve a replica-wide agreement on the transaction serialization order [18,24,17,23]. All these protocols adopt a single static strategy, unlike PolyCert which, not only allows for the simultaneous coexistence of multiple certification strategies, but autonomically determines, on a per-transaction basis, the most adequate replication protocol to employ using machine-learning techniques.

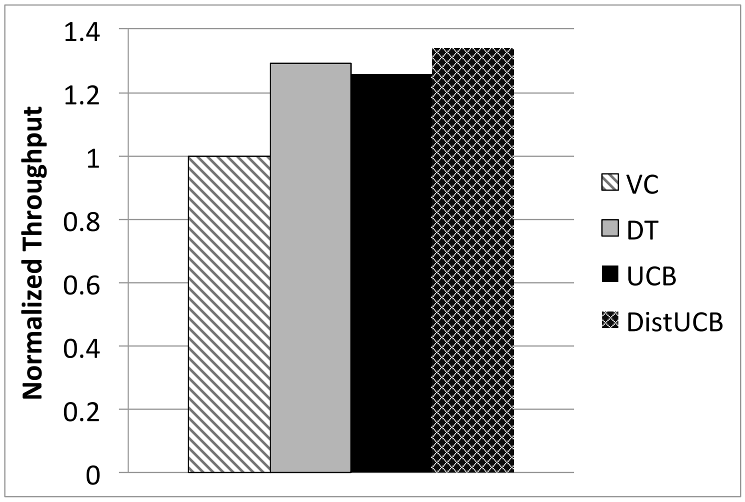

Fig. 8. Normalized throughput of the adaptive and VC protocols. NVC and BFC not reported as they caused the collapse of the GCS layer. (STMBench7 benchmark)

Machine learning techniques have already been used to predict the performance of computer systems in several contexts. These include works aiming at forecasting the throughput of TCP flows [22] and Pub-Sub systems [11], solutions aimed at automatically classifying traffic based on semi-supervised learning techniques [10], at automatizing the allocation of resources in cloud-computing infrastructures [33], or generating software aging models to be used in the context of rejuvenation frameworks [1]. Also, as noted in the text, the regressor decision tree oracle exploits our previous results in the area of machine-learning performance prediction of AB protocols, recently published in [9].

Our work is clearly related to the body of research on autonomic computing, and in particular to the field of self-optimizing databases. In this context, several approaches have been proposed based on the idea to automatically analyse the incoming workload, e.g. [20], to automatically identify the optimal database physical design or self-tune some of the DBMS inner management schemes, e.g. [5]. However, none of these approaches investigated the issues related to autonomically adapt the replication scheme. We argue that this is mainly due to the fact that current DBMSs, because of the high complexity of their architecture, lack the flexibility required to dynamically adapt such low level mechanisms.

7 Conclusions

Replication is of uttermost importance for in-memory NoSQL data platforms, which are emerging as an attractive alternative to conventional relational distributed databases. However, since the parameter space defining the workload of transactional applications is extremely wide, it is extremely challenging to devise universal transactional replication solutions capable of guaranteeing optimal performance in any possible scenario. In this paper we proposed, to the best of our knowledge for the first time in literature, a self-tuning adaptive scheme, which we named PolyCert, that allows for the simultaneous coexistence of multiple AB-based certification schemes. PolyCert uses parameter-free machine learning techniques to determine the optimal replication strategy to use on a per-transaction basis. The self-tuning strategy of PolyCert allows to achieve significant speed-ups when compared with non-adaptive certification protocols. Furthermore, it also improves the robustness of the replicated data platform, avoiding to saturate the GCS in the presence of transactions with extremely large read-sets, a main source of instability for several certification protocols.

References

1. Andrzejak, A., Silva, L.: Using machine learning for non-intrusive modeling and prediction of software aging. In: Proc. of the Network Operations and Management Symposium (NOMS). pp. 25–32. IEEE, Salvador de Bahia, Brazil (2008)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (2002)
3. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1986)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
5. Bruno, N., Chaudhuri, S.: An online approach to physical design tuning. In: Proc. of the International Conference on Data Engineering (ICDE). pp. 826–835 (2007)
6. Cachopo, J., Rito-Silva, A.: Versioned boxes as the basis for memory transactions. Science Computer Programming 63(2), 172–185 (2006)
7. Carvalho, N., Romano, P., Rodrigues, L.: Asynchronous lease-based replication of software transactional memory. In: Proc. of the International Middleware Conference (MIDDLEWARE). pp. 376–396. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2010)
8. Couceiro, M., Romano, P., Carvalho, N., Rodrigues, L.: D2STM: Dependable distributed software transactional memory. In: Proc. of the Pacific Rim International Symposium on Dependable Computing (PRDC). pp. 307–313. Shanghai, China (2009)
9. Couceiro, M., Romano, P., Rodrigues, L.: A machine learning approach to performance prediction of total order broadcast protocols. In: Proc. of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO). pp. 184 –193. Budapest, Hungary (2010)
10. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Offline/realtime traffic classification using semi-supervised learning. Performance Evaluation 64(9-12), 1194–1213 (2007)
11. Garces-Erice, L.: Admission control for distributed complex responsive systems. In: Proc. of the International Symposium on Parallel and Distributed Computing (ISPDC). pp. 226–233. IEEE Computer Society, Washington, DC, USA (2009)
12. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proc. of the International Conference on Management of Data (SIGMOD). pp. 173–182. ACM, New York, NY, USA (1996)
13. Guerraoui, R., Kapalka, M., Vitek, J.: STMBench7: a benchmark for software transactional memory. SIGOPS Operating Systems Review 41(3), 315–324 (2007)
14. Guerraoui, R., Rodrigues, L.: Introduction to Reliable Distributed Programming. Springer (2006)

15. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. The Journal of Machine Learning Research 3, 1157–1182 (2003)
16. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA (1994)
17. Kemme, B., Alonso, G.: Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication. In: Proc. of the Very Large Data Base Conference (VLDB). pp. 134–143. ACM, Cairo, Egypt (2000)
18. Kemme, B., Alonso, G.: A suite of database replication protocols based on group communication primitives. In: Proc. of the International Conference on Distributed Computing Systems (ICDCS). p. 156. IEEE Computer Society (1998)
19. Martin, M., Blundell, C., Lewis, E.: Subtleties of transactional memory atomicity semantics. IEEE Computer Architecture Letters 5(2), 17 (2006)
20. Martin, P., Elnaffar, S., Wasserman, T.: Workload models for autonomic database management systems. In: Proc. of the International Conference on Autonomic and Autonomous Systems (ICAS). p. 10. IEEE Computer Society, Washington, DC, USA (2006)
21. Miranda, H., Pinto, A., Rodrigues, L.: Appia, a flexible protocol kernel supporting multiple coordinated channels. In: Proc. of the International Conference on Distributed Computing Systems (ICDCS). pp. 707–710. IEEE, Phoenix, Arizona (2001)
22. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to TCP throughput prediction. In: Proc. of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS). pp. 97–108. ACM, New York, NY, USA (2007)
23. Patino-Martínez, M., Jiménez-Peris, R., Kemme, B., Alonso, G.: Scalable replication in database clusters. In: Proc. of the International Conference on Distributed Computing (DISC). pp. 315–329. Springer-Verlag (2000)
24. Pedone, F., Guerraoui, R., Schiper, A.: The database state machine approach. Distributed and Parallel Databases 14(1), 71–98 (2003)
25. Quinlan, J.R.: Cubist. http://www.rulequest.com/cubist-info.html
26. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
27. Robbins, H.: Some aspects of the sequential design of experiments. Bulletin of the American Mathematical Society 58(5), 527–35 (1952)
28. Romano, P., Rodrigues, L., Carvalho, N., Cachopo, J.: Cloud-TM: Harnessing the cloud with distributed transactional memories. SIGOPS Operating Systems Review 44, 1–6 (2010)
29. Schiper, N., Sutra, P., Pedone, F.: P-Store: Genuine partial replication in wide area networks. In: Proc. of the Symposium on Reliable Distributed Systems (SRDS). pp. 214–224. IEEE Computer Society, Washington, DC, USA (2010)
30. Schneider, F.B.: Replication management using the state-machine approach. ACM Press/Addison-Wesley Publishing Co. (1993)
31. Shevade, S.K., Keerthi, S.S., Bhattacharyya, C., Murthy, K.R.K.: Improvements to the SMO algorithm for SVM regression. IEEE Transactions on Neural Networks 11(5), 1188 –1193 (2000)
32. Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N., Helland, P.: The end of an architectural era: (it's time for a complete rewrite). In: Proc. of the International Conference on Very large Data Bases (VLDB). pp. 1150–1160. VLDB Endowment (2007)
33. Xu, J., Zhao, M., Fortes, J., Carpenter, R., Yousif, M.: Autonomic resource management in virtualized data centers using fuzzy logic-based approaches. Cluster Computing 11(3), 213–227 (2008)