Report on the 1st Euro-TM School on Transactional Memory - 6th School on Hot Topics in Distributed Computing (HTDC)

Maria Couceiro and Paolo Romano

17 March - 22 March 2013

Contents

Abstract

This documents reports the activities performed in the context of the 1st Euro-TM Doctoral School on Transactional Memory, which took place in La Plagne (FR) from March 17 to March 22 2013.

The school has been co-located with one of most renowned European doctoral schools on distributed computing, namely the PhD School on Hot Topics in Distributed Computing (HTDC). The high visibility of HTDC, together with a prestigious selection of lectures composed by widely known scientists, have been key to guarantee the success of the event.

The school had 50 participants (doctoral students) from 15 countries, and 6 lecturers from some of the leading academic and industrial institutions (2 of whom from non-COST countries).

The program has been designed to address a broad range of interdisciplinary topics. Three lectures were dedicated to analyse state of the art on different aspects of TM, namely its theoretical foundations (Prof. P. Fatouru, University of Crete, Greece), hardware supports (Prof. Christos Kozyriakis, Stanford University, USA) as well as algorithms, language features and implementation (Dr. Mark Moir, Oracle Labs, USA). Three lectures were instead devoted to some of the hottest topics in the areas of distributed computing and software engineering, namely self-adaptive systems (Prof. Carlo Ghezzi, Politecnico di Milano, IT), Byzantine Fault Tolerance (Dr. Marco Vukolic, Eurecom, FR) and data center management (Dr. Paolo Costa, Microsoft Research, UK). By combining TM-focused lectures with talks on some of the currently most relevant topics in distributed computing, the school brought together experts and students having different backgrounds and expertise, creating an ideal environment for interdisciplinary networking.

Finally, students have been provided ample opportunities to present their current research activities and gather feedback both from the lecturers of the school and from the other students. The students' presentations were organized in 3 doctoral sessions, hosting a total of 28 presentations that gave raise to lively discussions and triggered several new collaborations.

List of Participants

Speakers

Paolo COSTA
Panagiota FATOUROU
Carlo GHEZZI
Christos KOZYRAKIS
Mark MOIR
Marko VUKOLIC

Participants

Mohammad ALAGGAN
Valter BALEGAS
Gautier BERTHOU
Heverson BORBA RIBEIRO
Nicolas BRAUD-SANTONI
Stefan BRENNER
Victor BUSHKOV
Bapi CHATTERJEE
Tyler CRAIN
Frey DAVIDE
Jrmie DECOUCHANT
Carole DELPORTE
Amadou DIARRA
Diego DIDONA
Nuno DIEGUES
Joscha DRECHSLER
Swan DUBOIS
El Mahdi EL MHAMDI
Hugues FAUCONNIER
Carlo GHEZZI
George GIAKKOUPIS
Rachid GUERRAOUI
Lisong GUO
Ugur GREL
Sandeep HANS
Yarco HAYDUK
Arnaud JEGOU
Eleni KANELLOU
Anne-Marie KERMARREC
Tadeusz KOBUS
Maciej KOKOCINSKI
Jan KONCZAK
Lazaros KOROMILAS
Mihai LETIA
Giuliano LOSA
Jean-Pierre LOZI
Darko MAKRESHANSKI
Ioannis MANOUSAKIS
Erdal MUTLU
Nhan NGUYEN
Yiannis NIKOLAKOPOULOS
Burcu OZKAN
Nicolas PALIX
Matej PAVLOVIC
Luis PINA
Vivien QUEMA
Antoine RAULT
Srivatsan RAVI
Hugo RITO
Luiz Antonio RODRIGUES
Paolo ROMANO
Olivier RUAS
Iosif SALEM
Ge SONG
Julien SOPENA
Diogo SOUSA
Julien STAINER
Radu TUDORAN
Tiago VALE
Alexandre VAN KEMPEN
Jean-Francois VERDONCK
Paraskevas YIAPANIS

Program

Monday, March 18th

09:00 - 12:00	**Transactional memory algorithms, language features, and implementation** Mark Moir *(Oracle Labs)*
17:15 - 19:45	**BFT Systems** Marko Vukolic *(Eurecom)*

Tuesday, March 19th

09:00 - 12:00	**Hardware Support for Transactional Memory: the Debut** Christos Kozyrakis *(Stanford University)*
17:15 - 19:45	**Doctoral Session**

Wednesday, March 20th

09:00 - 12:00	**Dynamically Evolving Self-Adaptive Software** Carlo Ghezzi *(Politecnico di Milano)*
17:15 - 19:45	**Doctoral Session**

Thursday, March 21st

09:00 - 12:00	**Theory results in Transactional Memory** Panagiota Fatourou *(University of Crete)*
17:15 - 19:45	**Doctoral Session**

Friday, March 22nd

09:00 - 12:00	**How to Ski with Only One Ski or Data Centers and the Art of Doing More with Less** Paolo Costa *(Imperial College London)*

Monday, March 18th

4.1 Invited Speaker: Mark Moir

Title: Transactional memory algorithms, language features, and implementation

Abstract: I will discuss a variety of ways in which transactional memory can be used to improve on non-TM-based concurrent algorithms and data structures in terms of performance, scalability, simplicity, and usability. I will also discuss proposed transactional language features for C++, highlight some of the tradeoffs and challenges involved in specifying and implementing them, and describe some of the alternatives for implementing these features using hardware and/or software transactional memory.

Bio: Consulting Member of Technical Staff. Oracle Labs, United States
Mark Moir received the B.Sc.(Hons.) degree in Computer Science from Victoria University of Wellington, New Zealand in 1988, and the Ph.D. degree in Computer Science from the University of North Carolina at Chapel Hill, USA in 1996. He was an assistant professor in the Department of Computer Science at the University of Pittsburgh from 1996 until 2000. He then joined Sun Labs and subsequently formed the Scalable Synchronization Research Group, of which he is the Principal Investigator (the group is now part of Oracle Labs). He was named a Sun Microsystems Distinguished Engineer in 2009.
Dr. Moir's main research interests concern practical and theoretical aspects of concurrent, distributed, and real-time systems, particularly hardware and software support for programming constructs that facilitate scalable synchronization in shared memory multiprocessors.

4.2 Invited Speaker: Marko Vukolic

Title: BFT Systems

Abstract: Designing and implementing Byzantine fault tolerant (BFT) systems is complex. In this tutorial we will zoom into some of the major techniques and abstractions for BFT systems design and implementation with a particular focus on two fundamental applications: read/write storage and replication. Finally, we overview main impediments to using BFT in practice and briefly discuss the directions to circumventing those in future.

Bio: Marko Vukolic is an assistant professor in the Networking and Security department at Eurecom, France. He received his engineering degree in Communication Systems from University of Belgrade, Serbia in 2001 and his doctor of science degree in Distributed Systems from EPFL, Switzerland in

2008. He was affiliated with IBM Research Zurich where he spent time in the Storage Systems group as a post-doc from 2008 to 2010.

Tuesday, March 19th

5.1 Invited Speaker: Christos Kozyrakis

Title: Hardware Support for Transactional Memory: the Debut

Abstract: The end of single-thread performance scaling made it necessary to investigate technologies that can reduce the complexity of parallel programming for multi-core chips. Transactional Memory (TM) quickly emerged as one of the most promising technologies towards this goal. With TM, programmers simply declare that code blocks operating on shared data should execute as atomic and isolated transactions with respect to all other code. Concurrency control as multiple transactions execute in parallel is the responsibility of the system. This talk will focus on hardware support for transactional execution which is now finding its way into commercial multi-core chips. Hardware is necessary in order to address the significant overheads of software TM implementations and provide practical functionality without surprising results for common coding patterns. We will review the two general approaches for hardware support for TM, discuss their interface to software, and show that they are practical within the scope of modern multi-core chips. Moreover, we will show that the hardware mechanisms for TM can also support useful features for challenges beyond concurrency control, such as availability, security, and debugging. Finally, we will discuss the tradeoffs in the commercial implementations of hardware support for transactional execution.

Bio: Christos Kozyrakis is an Associate Professor of Electrical Engineering & Computer Science at Stanford University. He works on architectures, runtime environments, and programming models for parallel computing systems. At Berkeley, he developed the IRAM architecture, a novel media-processor system that combined vector processing with embedded DRAM technology. At Stanford, he co-led the Transactional Coherence and Consistency (TCC) project at Stanford that developed hardware and software mechanisms for programming with transactional memory. He also led the Raksha project, that developed practical hardware support and security policies to deter high-level and low-level security attacks against deployed software. Dr. Kozyrakis is currently working on hardware and software techniques for next-generation data centers. He is also a member of the Pervasive Parallelism Lab at Stanford, a multi-faculty effort to make parallel computing practical for the masses. Christos received a BS degree from the University of Crete (Greece) and a PhD degree from the University of California at Berkeley (USA), both in Computer Science. He is the Willard R. and Inez Kerr Bell faculty scholar at Stanford and a senior member of the ACM and the IEEE. Christos has received the NSF Career Award, an IBM Faculty Award, the Okawa Foundation Research Grant, and a Noyce Family Faculty Scholarship.

5.2 Doctoral Session

Name: Victor Bushkov
Title: On the Liveness-Safety Exclusion in Shared Memory Systems
Abstract: The correctness of a concurrent algorithm is expressed through safety and liveness properties. A safety property is defined as a prefix-closed and limit- closed set of well-formed execution histories of I/O automata that allows any inputs (invocations). A liveness property is defined as a set of infinite execution histories of I/O automata that permits any finite well-formed history, i.e. for every finite history there exists a continuation of that history in the liveness property. Because a liveness property basically states that some "good events eventually happen whilst a safety property states that some "bad events will never happen, it occurs sometimes that a safety property makes it impossible to guarantee a liveness property. We study, in a general and abstract manner, the inherent tradeoff between liveness and safety. Ideally, given a safety property S, we would like to determine the strongest liveness property that does not exclude S (i.e., that can be implemented with S), and respectively determine the weakest liveness property that excludes S (i.e., conflicts with S).
We say that a safety property excludes a liveness property of a shared object if every implementation of that object which ensures the safety property violates the liveness property. We give a necessary and sufficient condition when there is a strongest liveness property implementable with a given safety property, and a necessary and sufficient condition when there is a weakest liveness property that excludes a given safety property. The first condition states that there is a strongest liveness property that does not exclude safety property S iff liveness property $L_{max} \cup S_{block}$ does not exclude S; where L_{max} is the strongest liveness property and S_{block} is the subset of executions in which no response is allowed by S. The second condition states that there is a weakest liveness property that excludes safety property S iff for any two adversaries w.r.t. L_{max} the intersection of their behaviors is also a behavior of an adversary w.r.t. L_{max}; where an adversary w.r.t. L_{max} plays against an implementation to have it violate L_{max}.
For all common safety properties of shared memory systems $S_{block} = \emptyset$. Furthermore, there are pairs of adversaries w.r.t. L_{max} of which intersection is not an adversary w.r.t. L_{max}. Therefore, for common safety properties of shared objects for which there is no implementation ensuring L_{max} (e.g. consensus from registers or transactional memory), the above conditions can be considered as impossibility results which basically state that there is no weakest liveness property that excludes a given safety property (resp. strongest liveness property that does not exclude a given safety property).

Name: Tyler Crain
Title: A simple non-blocking hash table
Abstract: In this talk I will present an efficient, scalable, non-blocking (lock-free) hash-table. Hash tables implementing the map abstraction are a very popular and useful way to share data in a concurrent program. Previous efficient hash-tables have either used locks or were complex enough that they

were difficult to reason about. The algorithm presented here is a fairly simple and based off of the ConcurrentHashMap from the java.util.concurrent library. Each operation (including the rehash operation, which happens per-bucket) completes in a single compare-and-swap, preventing the need for complex helping mechanisms needed to ensure progress.

Name: Diego Didona
Title: Transactional AutoScaling
Abstract: Distributed software transactional memory are emerging as a powerful paradigm for distributed programming. By relying on the abstraction of the "transaction, the programmer is relieved not only of the burden of explicitly dealing with con- current access to shared data, but also of coping with failures and distributed synchronization processes needed to maintain data coherence. The effectiveness of the DSTM paradigm, however, is ultimately dependent on the workload exhibited by the application (e.g., conflict on data), on parameters of the underlying infrastructure (e.g., network latencies relevant to the synchronization phases), as well as on parameters of the TM layer itself (e.g., concurrency control scheme). The resulting multidimensional design space of a DSTM makes a priori determining the TM configuration which best fits the application's characteristics hard and time consuming. This task becomes impossible when considering dynamic workloads that change over time, thus requiring the identification and development of methodologies for determining, at runtime, the TM configuration which delivers the best performance.
My research has been focused on developing models which are able to forecast the effect of elastically scaling a DSTM application, i.e., changing the number of nodes/threads it is deployed over[1]. Such models take into account both workload's characteristics and platform-dependent parameters and are based on the joint exploitation of analytical modeling and machine learning. The former technique is exploited to forecast the effect of data contention with the adopted concurrency control scheme and replication protocol on performance. The latter is exploited to build a statistical model aimed at forecasting the impact on performance due to shifts in the utilization of system level resources (e.g,. network latencies) imputable to variations of the systems scale.
Lately I have been exploring the possibility of exploiting a hybrid approach based on modeling and on-line exploration[2]. At runtime, different configurations of the number of threads running on each node of the DSTM are explored, in order to gather feedbacks on the model's accuracy varying the degree of concurrency in the system. My current research is focused on devising mechanisms to incorporate these feedbacks into a self-correcting model, capable of increasing its accuracy on the basis of the performed explorations.
1. D. Didona, P. Romano, S. Peluso, F. Quaglia Transactional Auto Scaler: Elastic Scaling of In-Memory Transactional Data Grids In Proc. of the International Conference on Autonomic Computing (ICAC), 2012.
2. D. Didona, P. Felber, D. Harmanci, P. Romano, J. Schenker Elastic Scaling for Transactional Memory: From Centralized to Distributed Architectures Workshop on Hot Topics in Parallelism (HotPar), 2012.

Name: Nuno Diegues
Title: Enhancing Permissiveness in TM via Time-Warp
Abstract: The notion of permissiveness [1] in Transactional Memory (TM) translates to only aborting a transaction when it cannot be accepted in any history that guarantees a target correctness criterion. Unfortunately, this desirable property is often neglected by state of the art TMs [2,3,4,5]. In this context, existing literature on TM has highlighted an inherent trade-off between the efficiency achievable by an implementation of a TM algorithm, and the number of spurious aborts it introduces [1]. Indeed, in order to maximize implementation's efficiency, most TMs resort to aborting transactions under overly conservative conditions and, hence, incur in the risk of rejecting a significant number of serializable histories.
In this talk we will describe Time-Warp Multi-version (TWM), a novel multi-version concurrency control algorithm that aims to strike an unexplored balance between permissiveness and efficiency. TWM is based on the key idea of allowing a write transaction that has performed stale reads (i.e. missed the writes of concurrently committed transactions) to be safely serialized by "committing it in the past", which we call a time-warp commit. At its core TWM uses a novel, lightweight validation mechanism that can be implemented with minimum bookkeeping and computational overheads, thus ensuring efficiency. The key idea is to exclusively track some anti-dependencies developed by a committing transaction, hence avoiding onerous validation of the entire conflicts' graph (unlike algorithms that ensure permissiveness [6]). Another noteworthy feature of TWM is that it preserves what is probably the most important property of MVCC algorithms: the ability to ensure that read-only transactions can never be aborted (a property known as mv-permissiveness [7]).
We provide an extensive analysis of the correctness properties of TWM, demonstrating that it guarantees Virtual World Consistency [7], a safety property (strictly stronger than serializability) that is deemed as particularly relevant in the context of TM. We have also implemented a highly optimized, lock-free version of the TWM algorithm and compared it with two state of the art TMs using three standard benchmarks. Our evaluation study demonstrates the practicality of the TWM algorithm by highlighting consistent performance improvements across all considered workloads: TWM achieves 98% average speed up, with gains extending up to 7x in high concurrency scenarios. These results are a consequence of the drastic reduction of aborts achieved by TWM with respect to less permissive algorithms.
1. Rachid Guerraoui, Thomas A. Henzinger, and Vasu Singh. Permissiveness in transactional memories. In Proceedings of the 22nd international symposium on Distributed Computing, DISC 08, pages 305319, Berlin, Heidelberg, 2008. Springer-Verlag.
2. Pascal Felber, Christof Fetzer, and Torvald Riegel. Dynamic performance tuning of word-based software transactional memory. In Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, PPoPP 08, pages 237246, New York, NY, USA, 2008. ACM.
3. Dave Dice, Ori Shalev, and Nir Shavit. Transactional locking ii. In Proceedings of the 20th international conference on Distributed Computing, DISC06, pages 194208, Berlin, Heidelberg, 2006. Springer-Verlag.
4. Aleksandar Dragojevic, Rachid Guerraoui, and Michal Kapalka. Stretching

transactional memory. In Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation, PLDI 09, pages 155165, New York, NY, USA, 2009. ACM.
5. Sergio Miguel Fernandes and Joao Cachopo. Lock-free and scalable multi-version software transactional memory. In Proceedings of the 16th ACM symposium on Principles and practice of parallel programming, PPoPP 11, pages 179188, New York, NY, USA, 2011. ACM.
6. Hany E. Ramadan, Indrajit Roy, Maurice Herlihy, and Emmett Witchel. Committing conflicting transactions in an stm. In Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming, PPoPP 09, pages 163172, New York, NY, USA, 2009. ACM.
7. Dmitri Perelman, Rui Fan, and Idit Keidar. On maintaining multiple versions in stm. In Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing, PODC 10, pages 1625, New York, NY, USA, 2010. ACM.
8. Damien Imbs and Michel Raynal. Virtual world consistency: A condition for stm systems (with a versatile protocol with invisible read operations). Theoretical Computer Science, 444(0):113127, 2012.

Name: Sandeep Hans
Title: Abstractions for Transactional Memory
Abstract: Transactional memory (TM) has been hailed as a paradigm for simplifying concurrent programming. While several consistency conditions have been suggested for TM, they are stated as conditions on possible histories that implementations may export. This leaves a striking disconnect with programming language abstractions providing atomic blocks to the programmer.
We try to find the connection between the observations exported by histories satisfying TM consistency conditions and those expected with a simple programming abstraction using atomic blocks. We prove that opacity is a necessary and sufficient condition for providing observational refinement that preserves the views of every thread.
These results provide a new way to evaluate and compare TM consistency conditions. Further, they also reduce the effort of verifying that a TM satisfies the language abstractions, relying instead on proving only that it ensures opacity, as was already done for many TM implementations.

Name: Arnaud Jegou
Title: Privacy in distributed recommender systems
Abstract: We are interested about the privacy issues in distributed recommender systems. A distributed recommender system is a system providing users with recommendations of any type (movies, news...) in a distributed way. A typical way to implement such system is to use collaborative filtering, where users express their opinions about items, and these opinions are used to filter content and do relevant recommendations for the other users. One of the problem with these systems is privacy. A user wanting to use the system does not necessarily want other people to know what he or she is interested in,

especially in the case of sensitive subjects. We propose a mechanism to provide users with an improved level of privacy by disclosing only a subset of the user's interests with the addition of some randomization.

Name: Eleni Kanellou
Title: Glass half empty or glass half full? Pessimism VS Liveness in Transactional Memory
Abstract: One important aim of Transactional Memory is to make parallel programming easier and one topic of interest in this respect is how to hide eventual aborts and their appropriate handling from the programmer. Although in some cases, a programmer might want to take control of how to handle a transaction that has aborted, perform specific actions or abandon it instead of restart it, it can nevertheless be expected that in many situations, handling an aborted transaction may consist simply in restarting it. We are interested in methodologies which provide this functionality and we examine the possibilities of enhancing them with improved progress guarantees.

Name: Alexandre Van Kempen
Title: Efficient repair of erasure-coded data
Abstract: Classical erasure codes, e.g. Reed-Solomon codes, have been acknowledged as an efficient alternative to plain replication to reduce the storage overhead in reliable distributed storage systems. Yet, such codes experience high overhead during the maintenance process. In this paper we propose a novel erasure-coded framework tailored for networked storage systems. Our approach relies on the use of random codes coupled with a clustered placement strategy. This allows us to achieve efficient repair at the granularity of multiple files (hosted on the faulty machine). Our repair protocol leverages network coding techniques to reduce by half the amount of data transferred during maintenance, as several files are repaired simultaneously. This approach, as formally proven and demonstrated by our evaluation on a public experimental testbed, dramatically decreases both the bandwidth overhead during the maintenance process and the time to repair data lost upon failure. In addition, the implementation is made as simple as possible, aiming at a deployment into practical systems.

Name: Paraskevas Yiapanis
Title: Optimizing Software Runtime Systems for Speculative Parallelization
Abstract: Thread-Level Speculation (TLS) overcomes limitations intrinsic with conservative compile-time auto-parallelizing tools by extracting parallel threads optimistically and only ensuring absence of data dependence violations at runtime.
A significant barrier for adopting TLS (implemented in software) is the overheads associated with maintaining speculative state. Based on previous TLS limit studies we observe that on future multi-core systems we will likely have

more cores idle than those which traditional TLS would be able to harness. This implies that a TLS system should focus on optimizing for small number of cores and find efficient ways to take advantage of the idle cores. Furthermore, research on optimistic systems has covered two important implementation design points: eager vs. lazy version management. With this knowledge, we propose new simple and effective techniques to reduce the execution time overheads for both of these design points. This article describes a novel compact version management data structure optimized for space overhead when using a small number of TLS threads. Furthermore we describe two novel software runtime parallelization systems that utilize this compact data structure. The first software TLS system, MiniTLS, relies on eager memory data management (in-place updates) and, thus, when a mispeculation occurs a rollback process is required. MiniTLS takes advantage of the novel compact version management representation to parallelize the rollback process and is able to recover from misspeculation faster than existing software eager TLS systems.
The second one, Lector (Lazy inspECTOR) is based on lazy version management. Since we have idle cores, the question is whether we can create "helper tasks to determine whether speculation is actually needed without stopping or damaging the speculative execution. In Lector, for each conventional TLS thread running speculatively with lazy version management, there is associated with it a lightweight inspector. The inspector threads execute alongside to verify quickly whether data dependencies will occur. Inspector threads are generated by standard techniques for inspector/executor parallelization.
We have applied both TLS systems to seven Java sequential benchmarks, including three benchmarks from SPECjvm2008. Two out of the seven benchmarks exhibit misspeculations. MiniTLS experiments report average speedups of 1.8x for 4 threads increasing close to 7x speedups with 32 threads. Facilitated by our novel compact representation, MiniTLS reduces the space overhead over state-of-the-art software TLS systems between 96% on 2 threads and 40% on 32 threads. The experiments for Lector, report average speedups of 1.7x for 2 threads (that is 1 TLS + 1 Inspector threads) increasing close to 8.2x speedups with 32 threads (16+16 threads). Compared to a well established software TLS baseline, Lector performs on average 1.7x faster for 32 threads and in no case (x TLS +x Inspector threads) Lector delivers worse performance than the baseline TLS with the equivalent number of TLS threads (i.e. x TLS threads) nor doubling the equivalent number of TLS threads (i.e. x + x TLS threads).

Name: Yarco Hayduk
Title: Speculative Message Processing with Transactional Memory in the Actor Model
Abstract: Parallelism can be supported by the means of message passing and/or shared memory depending on the type of the underlying architecture: distributed or multi-core systems. To simplify the developer's task, it is desirable to provide programming abstractions that support both approaches.The "actor" model has been successfully used for scalable computing in distributed systems.Actors are objects with a local state, which can only be modified by the exchange of messages. One of the fundamental principles of actor models is to guarantee sequential message processing, which avoids typical concurrency

hazards.
A potential problem of this approach is that it limits the achievable message throughput, and therefore scalability on modern multi-core architectures. Preserving the sequential semantics of the actor model is, however, necessary for program correctness.
We propose to add support for speculative concurrent execution in actors using "transactional memory" (TM). The processing of each message is wrapped in a transaction executed atomically and in isolation, but concurrently with other messages, which allows us to scale while remaining compatible with the sequential semantics of the actor model. Further, our approach does not only support local processing of messages, but also coordinated processing on multiple actors.
We validate our design within the Scala programming language and the Akka framework. We show that the overhead of using transactions is hidden by the improved message throughput, thus leading to an overall performance gain. We further show that it is possible to switch from speculative to sequential message processing and identify cases where this switching improves efficiency.

Wednesday, March 20th

6.1 Invited Speaker: Carlo Ghezzi

Title: Dynamically Evolving Self-Adaptive Software

Abstract: Software is increasingly embedded in unstable settings where changes occur at all levels and continuously. Changes may occur at the requirements level. They may occur in the environment, and thereby affect the domain assumptions upon which the software was developed. They may also affect the computational infrastructure within which the software is run. Changes may lead an existing software into a situation where it fails to satisfy its intended goals. They may lead to failures or to unacceptable quality of service, and thus often to breaking the contract with the software's clients.
Software engineering has long studied the problem of off-line evolution (aka software maintenance). Many applications, however, are continuously running and ask for on-line change support as they are providing service: they require self-adapting capabilities. To achieve this goal, a paradigm shift is needed, which dissolves the traditional boundary between development time and run time. In particular, models must be kept at run-time and verification must be performed to detect possible requirements violations.
The lecture focused on the real-world requirements that lead to self-adaptive systems and then discussed how reflective capabilities can be designed to support self-adaptive capabilities. Emphasis was put on run-time verification (in the context of model checking) and on supporting safe dynamic software updates.
The work presented in the seminar has been funded by an Advanced Grant from the European Research Council (project SMScom).

Bio: Carlo Ghezzi is a Professor and Chair of Software Engineering. He has been the Rector's delegate for research, past member of the Academic Senate and of the Board of Governors, and past Department Chair. He held temporary or visiting positions at the University of California at Los Angeles, University of North Carolina at Chapel Hill, University of Padova, ESLAI-Argentina, University of California at Santa Barbara, Technical University of Vienna, University of Klagenfurt. He has an Adjunct Professor position at the Faculty of Informatics of the University of Lugano. He has been awarded the ACM SIGSOFT Distinguished Service Award. He has been on the evaluation board of several international research projects and institutions in Europe, Japan, and the USA. He is a regular member of the program committee of important conferences in the software engineering field, such as the ICSE and ESEC/FSE, for which he also served as Program and General Chair. He has given keynotes at several international conferences, including ESEC/FSE and ICSE. He has been the Editor in Chief of the ACM Trans. on Software Engineering and Methodology (from 2001 till 2006). He is currently an Associate Editor of IEEE Trans. on Software Engineering, Science of Computer Pro-

gramming, Service Oriented Computing and Applications, and Computing.
His research has been focusing on software engineering and programming languages. Currently, he is is especially interested in methods and tools to improve dependability of adaptable and evolvable distributed applications, such as service-oriented architectures and ubiquitous/pervasive computer applications. I co-authored over 180 papers and 8 books. I coordinated several national and international (EU funded) research projects and I have been awarded an Advanced Grant from the European Research Council.

6.2 Doctoral Session

Name: Tadeusz Kobus
Title: Hybrid Transactional Replication
Abstract: In our research we are interested in service replication schemes which allow increased service dependability and high performance. In particular, we investigate Transactional Replication (TR) which, beside providing fault tolerance and scalability, enables easy development of services by encompassing transactional semantics in common programming languages. As a tool for TR we use Distributed Transactional Memory (DTM), an extension of Transactional Memory [1] to distributed environment. Contrary to traditional approach to service replication based on State Machine (SM) [2], TR allows for parallel request processing (both on multicore hardware and in distributed environment) and greater code expressiveness with constructs for transaction rollback and retry. To precisely compare TR and SM we devised an analytical model which has been backed by experimental evaluation [3]. Surprisingly, neither of the schemes is superior - SM exhibits no parallelism thus does not scale while TR suffers under high contention. Therefore, the decision which scheme to choose has to be made on the premise of the expected workload. This unfortunately can be difficult. Moreover, the workload often changes over time. In order to overcome the limitations of both schemes we propose Hybrid Transactional Replication (HTR). A transaction is executed either optimistically by only one service replica in the deferred update mode (DU) [4], as typically in TR, or deterministically by all replicas in the state machine mode (SM). The choice is made dynamically by an oracle on per transaction execution basis. To choose the optimal execution mode the oracle uses multiple parameters such as transaction abort rate, duration of transaction's execution, network saturation, etc. This also means that the system is able to adapt to the changing workload. Not only HTR mitigates the performance limitations of both schemes, but it also introduces interesting modifications to DTM semantics itself. Thanks to the SM execution mode, a transaction can be guaranteed abort-free execution if necessary, thus allowing for irrevocable operations [5]. To our best knowledge our work is the first to tackle this issue in the context of Distributed TM. HTR is implemented in Paxos STM, our research DTM system. The obtained results clearly show the viability of this approach making this scheme a promising, versatile solution.
1. Nir Shavit and Dan Touitou. Software transactional memory. In Proc. of PODCS 95, August 1995.
2. Fred B. Schneider. Replication management using the state-machine approach, pages 169197. ACM Press/Addison-Wesley, 1993.

3. Pawel T. Wojciechowski, Tadeusz Kobus, and Maciej Kokocinski. Model-driven comparison of state-machine-based and deferred-update replication schemes. In Proc. of SRDS'12, October 2012.
4. Fernando Pedone, Rachid Guerraoui, and A.Schiper. The database state machine approach. Distributed and Parallel Databases, 14(1), July 2003.
5. Adam Welc, Bratin Saha, and Ali-Reza Adl-Tabatabai. Irrevocable transactions and their applications. In Proc. of SPAA 08, June 2008.

Name: Maciej Kokocinski
Title: State Machine vs Transactional Replication
Abstract: Distributed Transactional Memory (DTM), an extension of Transactional Memory (TM) [1], can be used to replicate a service. The resulting approach called Transactional Replication (TR) provides fault tolerance and enables easy development of distributed applications through rich transactional semantics offered by common programming languages. DTM systems usually rely on optimistic concurrency control schemes [2] to take advantage of parallelism of multinode/multicore environment. However, DTM imposes non-negligible over- head on transaction execution due to costly certification against conflicts between concurrent transactions and distribution of transactions' updates. Is TR a viable solution and how does it compare to traditional approaches such as State Machine (SM) replication [3]? To answer these questions and to better understand the differences between SM and TR we devised an analytical model [4]. In our model we consider two kinds of workloads. In the first one, execution of the request/transaction takes longer than communication between the processes. In the second one, the time required for network communication dominates over the request/transaction execution time. The model accounts for updating and read-only requests, optimizations of the used communication protocol (e.g. batching in atomic broadcast) and conflict function between concurrent transactions. The model allowed to precisely define the upper and lower bounds on the time of executing n concurrent requests on N servers for SM and TR. It also showed the speedup of read-only optimization for both approaches and allowed to compare performance of SM and TR against non-replicated (fault-prone) service. Analytical comparison between SM and TR is backed by experimental evaluation. For tests we used JPaxos for SM and Paxos STM for TR. The comparison is fair since both the tools share the same implementation of MultiPaxos algorithm for broadcasting messages with total order. We tested both systems under various workloads using known benchmarks. Surprisingly, none of the approaches is a clear winner. Although SM does not allow parallel request processing, in some cases it performs remarkably well. Under certain workloads SM even outperforms TR, since TR has to deal with increasing number of conflicts between concurrent transactions. However, one has to remember that, contrary to SM, TR oers rich transactional semantics that allows to better control the execution flow of the request.
1. Nir Shavit and Dan Touitou. Software transactional memory. In Proc. of PODCS '95, August 1995.
2. Fernando Pedone, Rachid Guerraoui, and A.Schiper. The database state machine approach. Distributed and Parallel Databases, 14(1), July 2003.

3. Fred B. Schneider. Replication management using the state-machine approach, pages 169197. ACM Press/Addison-Wesley, 1993.
4. Pawel T. Wojciechowski, Tadeusz Kobus, and Maciej Kokocinski. Model-driven comparison of state-machine-based and deferred-updatereplication schemes. In Proc. of SRDS '12, October 2012.

Name: Jan Konczak
Title: Recovery in Paxos & Futures in TM
Abstract: During the 10 minutes I want first to summarize my work on recovery in Paxos, later on present a fresh idea of using Futures in (software) transactional memory.
As for recovery, I'm going to tell why until now stable storage has been used in Paxos algorithm, and why it's a burden preventing high performance. I'll to summarize when the stable storage use can be reduced and to what extent the reduction is possible.
The second part of my talk will present the idea of using Futures in transactional memory. Assuming the transactions are ordered and read-sets and write-sets are known, each transaction can be started immediately, and use either the data values produced by previous transaction (if available), or operate on the Future object (otherwise). When the transaction performs an operation with side effect (visible outside the TM system), the system must wait for dependant transactions to complete.

Name: Mihai Letia
Title: Predicting Scalability Using the Obstruction Degree
Abstract: When choosing a priori one of several algorithms for implementing an abstraction, programmers turn to complexity theory to predict which solution would perform best. In the sequential world, this approach usually measures the number of steps that the algorithm needs to perform. As the world turns toward multicore computers, the number of steps required to execute an operation is no longer the only concern, but instead algorithms should perform better when running on more cores, i.e. scale well. Because of this, algorithms that perform more steps but where processes interfere with each other less are sometimes preferred.
Most existing complexity measures for concurrent programs do not encompass locks, although a large part of today's programs are built this way, making them unable to compare a program using locks to an alternative that avoids them. Furthermore, they consider the worst case that the algorithm can encounter while executing, thus failing to give predictions for the common case, the one that typically arises during execution.
In this talk I present the notion of obstruction degree, a metric for predicting the scalability of concurrent algorithms before implementing them. We illustrate its simplicity and wide application range using various lock-based and lock-free algorithms, which we then split into classes of equivalence. We convey the accuracy of the scalability predictions made using the obstruction degree through experimental measures.

Name: Matej Pavlovic
Title: Framework for Fault-Tolerant Distributed Computation in a Dynamic Environment
Abstract: In the context of distributed computing, the assumption of a static network containing a fraction of faulty nodes has been very prevalent. However, many modern distributed applications are highly dynamic with respect to the composition of the network and must count on nodes joining and leaving the network very frequently. This project, based on the paper `On Dynamic Distributed Computing` by Rachid Guerraoui, Florian Huc and Anne-Marie Kermarrec, addresses this issue.
The framework being developed implements a new method to partition a network of nodes into clusters of small size and to create and maintain a random expander graph on these clusters. This is done in a randomized and fully distributed manner and guarantees that, with high probability, every cluster contains a majority of honest nodes in presence of a Byzantine adversary controlling a fraction of all nodes in the network. The desired properties of the created network structure (such as the majority of honest nodes in each cluster) are maintained despite a high level of churn induced by nodes joining and leaving the network. The network can expand polynomially with respect to its initial size by dynamically adding and removing nodes, while its desired properties are still guaranteed.
Clusters with the overlay[1] provide an abstraction over a network of single nodes, which allows reliable distributed computation despite some of the nodes malfunctioning. In this abstract network, a cluster can be considered a single reliable computational unit. The abstraction thus allows efficient solving of many hard problems in distributed computing such as Byzantine agreement, computation of an aggregate function, Byzantine broadcast (without the broadcasting node having full knowledge of the network) or peer sampling tolerating a Byzantine adversary.

Name: Antoine Rault
Title: Improving privacy in a decentralised news recommendation system with indirect similarity
Abstract: Nowadays, quantity and frequency of content generation on the web make it impossible for someone to manually filter out what is relevant to its interests. One solution to this issue is to automate the filtering using a recommendation system.
We focus on WhatsUp, a fully decentralised peer-to-peer system for recommending news, providing scalability and resistance to censorship. It uses collaborative filtering (CF) for news recommendation and gossip for disseminating news. However, decentralised CF requires peers (representing users) to exchange their opinions on news, which poses a privacy risk for users, especially

[1] The overlay is formed by the above-mentioned expander graph with vertices corresponding to clusters and edges corresponding to connections between clusters.

when it comes to politics or religion-related news.
In WhatsUp, peers have a continuously changing local view of the system thanks to random peer sampling (RPS). As peers use the system, their view changes from totally random peers to peers with similar interests, using a similarity metric. This presents a privacy risk because a peer needs to know the profile of interests of other peers to compute their similarity.
The key idea in this talk is to allow similarity computation between two peers without needing them to reveal their profile of interests to each other. We are working on allowing this by adapting the well-known cosine similarity into an indirect similarity metric. By this, we mean that two peers rather compute their similarity to a set of arbitrary profiles of interests, which are unique to each pair of peers by generating them from a seed obtained through a Diffie-Hellman exchange. We are getting encouraging first results with performances comparable to those of WhatsUp using standard cosine similarity.

Name: Hugo Rito
Title: GHOST: Green, History-aware, and Optimistic Software Transactions
Abstract: Software Transactional Memory (STM) [3] is one promising abstraction to simplify the task of writing highly parallel applications. However, STMs are still impractical in write-dominated workloads where frequent transaction reexecutions place a significant stress on the STM runtime that hinders performance.
Notwithstanding the problem of conflicts being related to STMs' optimistic approach to concurrency, I argue that an optimistic solution capable of dealing with conflicts efficiently can be more beneficial than a conservative approach that over limits concurrency between transactions. My belief is that the information collected during a transaction, such as the transaction's read-set and write-set, is crucial to improve STMs performance in conflict-intensive workloads as it allows transactions to learn from their past and adapt accordingly in the future.
Recently, and inspired by the idea that STMs should go green, I showed how an STM runtime can recycle [1] and reuse [2] per-transaction information to incorporate almost for free memoization into the STM runtime. As expected, memoization behaves the best when applied to long-running transactions composed of several semi-independent tasks which behavior remains unchanged between reexecutions. Performance wise, memo-transactions were up to 3 times faster than normal transactions in high-load and high-contention scenarios in a machine with 48 cores, but conflicted as much as non-memo transactions.
For that reason, I am now exploring how to reduce the number of conflicts using a smart scheduler that remembers which transactions usually conflict and prevents them to execute concurrently. Current scheduling solutions are too conservative and over-serialize transactions, so I intend to increase the parallelism between conflicting transactions by taking advantage of fine-grained information regarding which accesses cause more conflicts and at which point during the transaction's lifetime do these conflicting accesses happen.
1. Couceiro, M., Romano, P.: Where does transactional memory research stand and what challenges lie ahead?: WTM 2012, EuroTM workshop on transactional memory. SIGOPS Oper. Syst. Rev. 46(2), 8792 (Jul 2012)

2. Rito, H., Cachopo, J.: FlashbackSTM: Improving STM performance by remem- bering the past. In: 25th International Workshop on Languages and Compilers for Parallel Computing. Waseda University (September 2012)
3. Shavit, N., Touitou, D.: Software transactional memory. In: Proceedings of the 14th annual ACM symposium on Principles of Distributed Computing. pp. 204 213. PODC '95, ACM (1995)

Name: Julien Stainer
Title: Synchrony Weakened by Message Adversaries vs Asynchrony Enriched with Failure Detectors
Abstract: A message adversary is a daemon that suppresses messages in round-based message-passing synchronous systems in which no process crashes. A property imposed on a message adversary defines a subset of messages that cannot be eliminated by the adversary. It has recently been shown that when a message adversary is constrained by a property denoted TOUR (for tournament), the corresponding synchronous system and the asynchronous crash-prone read/write system have the same computability power for task solvability. This talk introduces new message adversary properties (denoted SOURCE and QUORUM), and shows that the synchronous round-based systems whose adversaries are constrained by these properties are characterizations of classical asynchronous crash-prone systems (1) whose communication is through atomic read/write registers or point-to-point message-passing, and (2) enriched with failure detectors such as and . Hence these properties characterize maximal adversaries, in the sense that they define strongest message adversaries equating classical asynchronous crash-prone systems. They consequently provide strong relations linking round-based synchrony weakened by message adversaries with asynchrony enriched with failure detectors. This not only enriches our understanding of the synchrony/asynchrony duality, but also allows for the establishment of a meaningful hierarchy of property-constrained message adversaries.

Name: Luis Pina
Title: Dynamic Software Updates using Software Transactional Memory
Abstract: Upgrading a program is unavoidable to correct bugs, add features, or improve performance. This is, however, a disruptive operation that involves stopping and restarting the running program. The ability to upgrade a program without stopping it is thus increasingly important in a world where service downtime directly maps to revenue loss.
There are, of course, high availability systems that simply cannot stop and already support dynamic upgrades. Such systems typically rely on redundant hardware, which is already present for fault tolerance, to upgrade some machines separately whilst others provide service. They employ complex and domain specific algorithms to migrate their state and restrict the upgrade's expressiveness to ensure safety or compatibility between versions, which means that a practical solution to perform dynamic software upgrades still eludes re-

searchers and engineers.
My work focuses on using a multi-versioned Software Transactional Memory (STM) [3] to allow software upgrades that occur atomically, concurrently with the execution of the program, and that convert the program's data lazily, as data is progressively accessed by the execution of the upgraded program. The multi-versioned STM (JVSTM [2]) plays a major role in our system because it allows to convert objects lazily, as they are needed by the application code, but still guarantee that the conversion code will get a consistent view of the program state at the logical time of the update [1], regardless of how long ago it took.

1. Boyapati,C.,Liskov,B.,Shrira,L.,Moh,C.H.,Richman,S.:Lazy modular upgrades in persistent object stores. In: Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. pp. 403417. OOPSLA 03, ACM, New York, NY, USA (2003), http://doi.acm. org/10.1145/949305.949341
2. Cachopo, J., Rito-Silva, A.: Versioned boxes as the basis for memory transactions. Sci. Comput. Program. 63, 172185 (December 2006), http://dl.acm.org/citation.cfm?id=1228561.1228566
3. Shavit, N., Touitou, D.: Software transactional memory. In: Proceedings of the 14th annual ACM symposium on Principles of Distributed Computing. pp. 204 213. PODC '95, ACM (1995)

Thursday, March 21st

7.1 Invited Speaker: Panagiota Fatourou

Title: Theory results in Transactional Memory

Abstract: This talk presents a collection of theory results in TM. Specifically, we will study several consistency and progress conditions, impossibility results and lower bounds, as well as design decisions for TM algorithms, some common TM implementations, relations of TM algorithms to universal constructions, computability, performance results and tradeoffs in TM Computing.

Bio: Panagiota Fatourou is an Assistant Professor at the Department of Computer Science, University of Crete and an affiliated faculty member of the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH). Prior to joining the University of Crete and FORTH ICS, she was a full-time faculty member at the Department of Computer Science of the University of Ioannina. The academic years 2000 and 2001, she was a postdoc at Max-Planck Institut fur Informatik, Saarbrucken, Germany, and at the Computer Science Department of the University of Toronto, Canada. She got a degree in Computer Science from the University of Crete, and a PhD degree in Computer Engineering from the University of Patras.
Her research interests focus on the design and analysis of algorithms and data structures, lower bounds and impossibility results in distributed and parallel computing, decentralized data structures, concurrent objects, synchronization protocols, communication protocols, internet protocols, peer-to-peer computing, parallel programming languages, multithreaded computing, scheduling and load balancing. P. Fatourou has published her research results in the most prestigious conferences (ACM STOC, ACM PODC, ACM SPAA) and journals (J. of the ACM, SIAM J. on Computing) of her field. She has participated in several projects funded by the EU and she is currently coordinating an EC-funded Marie Curie Initial Training Network, called TransForm. She has repeatedly served as a member of the PC of the most well-known conferences in her field and she is the editor of the Distributed Computing Column of the Bulletin of the European Association for Theoretical Computer Science.

7.2 Doctoral Session

Name: Mohammad Alaggan
Title: Assumption of Independence and Differential Privacy
Abstract: With the increase in the number of collaborative social platforms and the need to leverage the data in these systems for recommendation and personalization, there has also been the need to protect the privacy of the users' private profiles. One popular and recent solution is differential privacy [1], which have been applied to collaborative social platforms [2, 3]. However, recent analysis have shown that differential privacy is sensitive to the assump-

tion of independence among the items to protect [4]. If the items to provide privacy for are not independent, then differential privacy does not adequately limit inference about their existence in the users' profiles. The authors of [4] argued that the notion of evidence of participation is central to addressing this issue. Nonetheless, they do not provide a general definition, but rather construct their critique on specific and artificial examples. The main questions our research is trying to address include:
1. What is evidence of participation and how to measure it. And whether is could be defined without assumptions about the data generation process.
2. How is that relevant to real-life datasets (like Digg).
3. It is only claimed that independence is sufficient for differential privacy, but is it necessary? Can differential privacy survive some degree of dependence?
4. If differential privacy fails to provide adequate guarantees in case of dependence, can we measure the resulting breach, theoretically or experimentally? What would an experimental baseline be? What is the right measure?
The research is at a very preliminary phase and there are more questions than answers.
1. C. Dwork and M. Naor, On the Difficulties of Disclosure Prevention in Statistical Databases or the Case for Differential Privacy, Journal of Privacy and Confidentiality, vol. 2, no. 1, pp. 93107, 2010.
2. M. Alaggan, S. Gambs, and A.-M. Kermarrec, Private Similarity Computation in Distributed Systems: From Cryptography to Differential Privacy, in Proceedings of the 15th International Conference on the Principles of Distributed Systems (OPODIS'11), ser. Lecture Notes in Computer Science, A. F. Anta, G. Lipari, and M. Roy, Eds., vol. 7109. Toulouse, France: Springer, December, 2011, pp. 357377.
3. M. Alaggan, S. Gambs, and A. Kermarrec, BLIP: Non-Interactive Differentially-Private Similarity Computation on Bloom Filters, in Proceedings of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS12), Toronto, Canada, October, 2012.
4. D. Kifer and A. Machanavajjhala, No free lunch in data privacy, Proceedings of the 2011 International Conference on Management of Data (SIGMOD'11), p. 193, 2011.

Name: Gautier Berthou
Title: RAC: a Freerider-resilient, Scalable, Anonymous Communication Protocol
Abstract: Enabling anonymous communication over the Internet is crucial. The first protocols that have been devised for anonymous communication are subject to freeriding. Recent protocols have thus been proposed to deal with this issue. However, these protocols do not scale to large systems, and some of them further assume the existence of trusted servers. In this talk, I will present RAC, the first anonymous communication protocol that tolerates freeriders and that scales to large systems. Scalability comes from the fact that the complexity of RAC in terms of the number of message exchanges is independent from the number of nodes in the system. Another important aspect of RAC is that it does not rely on any trusted third party. We experimentally evaluate RAC

using simulations. Our evaluation shows that, whatever the size of the system (up to 100.000 nodes), the nodes participating in the system observe the same throughput.

Name: Nicolas Braud-Santoni
Title: Fast Byzantine Agreement
Abstract: We present the first probabilistic Byzantine Agreement algorithm whose communication and time complexities are poly-logarithmic. So far, the most effective probabilistic Byzantine Agreement algorithm had communication complexity $\tilde{O}(\sqrt{n})$ and time complexity $\tilde{O}(1)$. Our algorithm is based on a novel, unbalanced, almost everywhere to everywhere Agreement protocol which is interesting in its own right.

Name: Jérémie Decouchant
Title: Huge Pages in Multicore Systems
Abstract: Most general-purpose processors provide support for memory pages of large sizes, called huge pages in Linux systems. Huge pages enable each entry in the TLB to map a large physical memory region into a virtual address space. This dramatically increases TLB coverage, reduces TLB misses, and promises performance improvements for many applications.
However, with current Linux implementations of huge pages, applications may suffer an important performance degradation (up to 20%). This degradation is not well understood, and forbids a broader use of huge pages.
In this work, our aim is to understand the impact of huge pages, and to design a mechanism that automatically obtains the best from the variety of page sizes available nowadays.

Name: Lisong Guo
Title: Pinpointing the Offending Code in a Kernel Oops
Abstract: A kernel oops is the error message that Linux kernel dumps on the console and in system logs when it detects an error(Oops) in the execution of the kernel. A kernel oops contains first-hand information about the state of the kernel at the time of error, which is useful for debugging. Nevertheless, the task of debugging with a kernel oops is often non-trivial, even for its primary step—to pinpoint the line of code that caused the oops. The current standard approach involves recompiling the code and decoding with the debugging tool gdb to pinpoint the offending line of code, which is a largely manual process that requires substantial expertise from a developer. Furthermore, the approach imposes a series of hard-to-satisfy constraints on the execution and its performance suffers from several limitations. We propose a preliminary design of a novel approach to pinpoint the offending code automatically. Plus, we define a series of criteria for experiments to evaluate our approach, based on a data set extracted from a real-world repository of kernel oopses.

Name: Yiannis Nikolakopoulos
Title: A Study of the Behavior of Synchronization Methods in Commonly Used Languages and Systems
Abstract: Synchronization is a central issue in concurrency and plays an important role in the behavior and performance of modern programmes. Programming languages and hardware designers are trying to provide synchronization constructs and primitives that can handle concurrency and synchronization issues efficiently. Programmers have to find a way to select the most appropriate constructs and primitives in order to gain the desired behavior and performance under concurrency. Several parameters and factors affect the choice, through complex interactions among (i) the language and the language constructs that it supports, (ii) the system architecture, (iii) possible run-time environments, virtual machine options and memory management support and (iv) applications.
We present a systematic study of synchronization strategies, focusing on concurrent data structures. We have chosen concurrent data structures with different number of contention spots. We consider both coarse-grain and fine-grain locking strategies, as well as lock-free methods. We have investigated synchronization-aware implementations in C++, C# (.NET and Mono) and Java. Considering the machine architectures, we have studied the behavior of the implementations on both Intels Nehalem and AMDs Bulldozer. The properties that we study are throughput and fairness under different workloads and multiprogramming execution environments. For NUMA architectures fairness is becoming as important as the typically considered throughput property. To the best of our knowledge this is the first systematic and comprehensive study of synchronization-aware implementations.
This paper takes steps towards capturing a number of guiding principles and concerns for the selection of the programming environment and synchronization methods in connection to the application and the system characteristics.

Name: Joscha Drechsler
Title: Distributed Reactive Programming
Abstract: Reactive programming has proven effective in simplifying the implementation of applications that react to user input or other external events. This programming paradigm allows developers to express dependencies among reactive entities and enables a declarative style instead of requiring developers to write a lot of boilerplate code for Observer patterns. We envision a scenario in which these concepts are used as the fundamental abstractions to shape a distributed middleware based on remotely shared reactive values.

Name: Valter Balegas
Title: Providing Strong Consistency Semantics Efficiently in Geo-Replicated Databases
Abstract: Cloud infra-structures are a well established environment to deploy

worldwide services, such as the popular Facebook, Twitter or YouTube. Millions of users are connected to these services, who can be far from the host, increasing the operations latency. This problem can affect the user's experience and lead to the loss of clients [4]. The demand for scalability and low-latency led to the development of a large number of replicated databases [2, 3, 6-9, 12]. The trade-off is that data consistency is relaxed, exposing stale data to provide fast response [3, 6, 8, 9] or paying the price to communicate with a remote replica, when it is not possible to use the nearest [2, 12]. Authors frequently classify the consistency level of their systems. In one side of the spectrum we have strong consistency and, in the opposite side, we have eventual consistency. A partitionable system can provide either strong consistency or availability, as state by the CAP theorem [5]. Many systems provide consistency guarantee stronger than eventual consistency, providing a consistent view of the database despite the ongoing operations. Causal consistency [1] is a popular consistency level, often adopted [2, 9, 12] because it avoids many inconvenient operations ordering while preserving a great degree of concurrency. Unfortunately, concurrent operations in distributed systems can lead to divergence if no concurrency control mechanism is used, as operations can be executed concurrently over different replicas of the same objects. Systems that allow divergence have to apply some mechanism to reconcile replica's state, or expose conflicts to the application [13]. On the other hand, systems that preclude replica divergence require coordination.

In some cases, replica convergence cannot maintain database invariants. The bank account is a classical example to show how concurrent operations can affect the semantics of application's data. In a replicated banking system that does not allow accounts balance to go negative, two clients that have access to the same account can withdraw money concurrently from different replicas. Since none of the operations is aware of the other, the sum of the withdrawn money can exceed the accounts balance. Transactions serialization would be required to avoid this anomaly. However, serialization can incur in a huge latency penalty [2, 7, 12].

In this work, we intend to explore the idea of using reservations [11] to avoid the cost of serialization to preserve database invariants. Reservations consist in requesting rights to modify an object or a portion of an object similarly to locks. Clients are only allowed to modify what they reserved, or objects that do not require reservations. Reservations can be obtained outside of the transaction execution flow, thus transforming an operation that would require global coordination in a fast operation [7]. Ultimately, clients can operate disconnected over cached data and connect opportunely to commit the updated values, as long as the reservations remain valid. The bank account example is easy to solve using reservations and they can be used to maintain more complex data invariants.

We will explore reservations and experiment different strategies, to provide a geo-replicated database with strong consistency guarantees and low-latency transactions. In the following sections we will show how the idea of reservations can be applied to a counter data- type and enumerate some challenges that we will face in this project.

1. Mustaque Ahamad, Gil Neiger, James E. Burns, Prince Kohli, and Phillip W. Hutto. Causal memory: Definitions, implementation, and programming. Distributed Computing, 9(1):3749, 1995.

2. Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. Pnuts: Yahoo!s hosted data serving platform. Proc. VLDB Endow., 1(2):12771288, August 2008.
3. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, SOSP '07, pages 205220, New York, NY, USA, 2007. ACM.
4. J. Brutlag E. Schurman. Perforamnce related changes and their user impact. Presented at Velocity Web Performance and Operations Conference, June 2009.
5. Seth Gilbert and Nancy Lynch. Brewers conjecture and the feasibility of consistent, available, partitiontolerant web services. SIGACT News, 33(2):5159, June 2002.
6. Avinash Lakshman and Prashant Malik. Cassandra: structured storage system on a p2p network. In Proceedings of the 28th ACM symposium on Principles of distributed computing, PODC '09, pages 55, New York, NY, USA, 2009. ACM.
7. Cheng Li, Daniel Porto, Allen Clement, Johannes Gehrke, Nuno Preguica, and Rodrigo Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, OSDI'12, pages 265 278, Berkeley, CA, USA, 2012. USENIX Association.
8. Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Dont settle for eventual: scalable causal consistency for wide-area storage with cops. In Proceedings of the Twenty- Third ACM Symposium on Operating Systems Principles, SOSP '11, pages 401416, New York, NY, USA, 2011. ACM.
9. Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Stronger semantics for low-latency geo-replicated storage. 10th Symposium on Networked Systems Design and Implementation, 2013.
10. Patrick E. O'Neil. The escrow transactional method. ACM Trans. Database Syst., 11(4):405430, December 1986.
11. N. Preguica, C. Baquero, F. Moura, J. L. Martins, R. Oliveira, H. J. Domingos, J. O. Pereira, and S. Duarte. Mobile transaction management in mobisnap. In Proceedings of the 2000 ADBIS-DASFAA, number 1884 in Lecture Notes on Computer Science, pages 379386. Springer-Verlag, 09 2000.
12. Yair Sovran, Russell Power, Marcos K. Aguilera, and Jinyang Li. Transactional storage for geo- replicated systems. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP11, pages 385400, New York, NY, USA, 2011. ACM.
13. D. B. Terry, M. M. Theimer, Karin Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. In Proceedings of the fifteenth ACM symposium on Operating systems principles, SOSP 95, pages 172182, New York, NY, USA, 1995. ACM.

Name: Radu Tudoran

Title: Big Data Management and Processing in Clouds. Case study: A-Brain
Abstract: The emergence of cloud computing has brought the opportunity to use large-scale compute infrastructures for a broader and broader spectrum of applications and users. As the cloud paradigm gets attractive for the "elasticity in resource usage and associated costs (the users only pay for resources actually used), cloud applications still suffer from the high latencies and low performance of cloud storage services. As Big Data analysis on clouds becomes more and more relevant in many application areas, enabling high-throughput massive data processing on clouds becomes a critical issue. We address this challenge at the level of cloud storage, by introducing a concurrency-optimized data storage system (called TomusBlobs) which federates the virtual disks associated to the Virtual Machines running the application code on the cloud. We demonstrate the performance benefits of our solution for efficient data-intensive processing by building an optimized prototype MapReduce framework for Microsofts Azure cloud platform based on TomusBlobs, which is extended into hierarchical Geo-distributed MapReduce framework. Finally, we specifically address the limitations of state-of-the-art MapReduce frameworks for reduce-intensive workloads, by proposing MapIterativeReduce as an extension of the MapReduce model. We validate the above contributions through large-scale experiments with real-world applications on the Azure commercial cloud, using resources distributed across multiple data centers.

Friday, March 22nd

8.1 Invited Speaker: Paolo Costa

Title: How to Ski with Only One Ski or Data Centers and the Art of Doing More with Less

Abstract: Online and cloud services such as those run by Amazon, Google, Facebook and Microsoft are powered by data centers, large warehouses hosting hundreds of thousands of servers around the world. In this lecture, I will provide an overview of data centers, discussing several aspects related to energy, hardware, software and networking infrastructure. I will present some of the challenges occurring when deploying large-scale data centers and describe the solutions adopted by researchers and practitioners. Finally, I will conclude the lecture with a brief overview of CamCube, a recent research project in which we have been exploring alternative hardware and software designs for data centers. By providing applications with a more fine-grained control on network resources, CamCube enables increasing performance and reducing development complexity and cluster costs.

Bio: Paolo Costa is a full time researcher in the Systems and Networking Group of the Microsoft Research Lab in Cambridge and a senior research investigator with the Department of Computing of Imperial College London.
In the past, he had been a research faculty at Imperial College London and I had been a Postdoctoral Researcher in the Computer Systems group at Vrije Universiteit Amsterdam. He holds a M. Sc. and Ph.D. degree in Computer Engineering from the Politecnico di Milano.
His research interests lie at the intersection of systems and networking with particular emphasis on large-scale networked systems, ranging from sensor and mobile networks to overlay networks and data centers. His current research focuses on providing a better integration and synergy between applications and networks in data centers to improve performance and reduce complexity.