Project no.	034084
Project acronym:	SELFMAN
Project title:	*Self Management for Large-Scale Distributed Systems based on Structured Overlay Networks and Components*

European Sixth Framework Programme Priority 2, Information Society Technologies

Periodic Activity Report - Year Three

Due date of deliverable:	July 15, 2009
Actual submission date:	July 15, 2009
Start date of project:	June 1, 2006
Duration:	36 months
Dissemination level:	CO

Contents

1 Executive summary

Managing large distributed applications is extremely difficult to do with conventional technologies, and because of the continued growth and penetration of the Internet, it will soon become impossible. The SELFMAN project has built a solution: a scalable self-managing infrastructure that provides a replicated transactional storage as well as other services needed for self management. We have built two implementations of this storage, Scalaris and Beernet, together with their supporting technologies, the Kompics component model, the SicSim discrete event simulator, the MyP2PWorld network and concurrency emulator, and the Mozart Programming System. We have built a distributed media streaming product, PeerTV, that is being successfully commercialized.

In the third year of the project, we have thoroughly studied and evaluated the self-management mechanisms: self configuration, self tuning, self healing, and self protection. We have developed four applications: one commercial product, PeerTV, and three demonstrator applications, a Distributed Wikipedia, a distributed recommendation system (Sindaca), and a distributed graphic editor for mobile devices (DeTransDraw). We have evaluated the self-management mechanisms of these applications both qualitatively and quantitatively. We have studied self protection (security) for structured overlay networks. We find that self protection needs an appropriate network topology: Small-World Networks provide an advantageous topology compared to exponential structured overlay networks. Finally, we have built a complete self-configuration framework consisting of three libraries, WorkflOz (dynamic distributed workflows), FructOz (dynamic deployment and configuration) and LactOz (dynamic navigation and monitoring). From this experience, we have distilled a set of Guidelines that give the first presentation of a detailed methodology for building self-managing systems.

2 Project objectives and major achievements

The overall objective of SELFMAN is to address directly the challenge of building large distributed systems. The approach taken by SELFMAN is to build and prove the effectiveness of an application infrastructure for scalable self-managing applications based on extending a structured overlay network with self-* services and a transaction protocol.

We briefly recapitulate the project's first two years. In the first year, we built two structured overlay networks (DKS and P2PS) and we explored the programming model and component architecture needed for expressing their self-management operations. We also defined a transaction protocol and collected application scenarios. In the second year, we built our first transactional storage, Scalaris, and built two applications, a Distributed Wiki and a media streaming application (which would later become PeerTV). The Distributed Wiki won first prize in the IEEE SCALE 2008 contest. We also completed the design of the Kompics component model and made a first implementation. It has become the standard model for SELFMAN.

In the third year, all of this work has matured. We have made a second implementation of a transactional storage, called Beernet, on top of the recently released Mozart 1.4.0 system, and we ported Beernet to the Android operating system of the HTC Magic gPhone to support DeTransDraw, a collaborative drawing tool. We successfully released the PeerTV product and we built a demonstrator of a recommendation system, Sindaca. We have made detailed qualitative and quantitative evaluations of three demonstrators, namely the Distributed Wiki, PeerTV, and DeTransDraw. We have worked on the four main self-* services and how to combine them. We have distilled this work into a first methodology for building self-managing applications.

2.1 Major achievements in the third year

In Year 3, we have achieved the following main results:

- *Transactional store.* We made a public release of the first open-source version of Scalaris (transactional key-value store). The complete code is available under the Apache License at `scalaris.googlecode.com`.
- *Distributed media streaming.* We made the PeerTV product for distributed media streaming. Peerialism is currently in the process of being acquired by GGF for the PeerTV technology (see Section 4.2.1).

- *Component model.* We made a public release of the Kompics implementation at `kompics.sics.se`. A Peer-to-Peer component framework was implemented together with tools for evaluating P2P systems. Real system implementations (in Kompics) can now be executed both in reproducible simulation and real-time emulation, locally, or in distributed deployment. KTH(P2) validated the Kompics component framework by using it to prepare and support lab assignments in two courses on distributed systems: Advanced Course in Distributed Systems (ID2203) and Distributed Systems, Peer-to-Peer and Grids (ID2210).

- *Qualitative and quantitative assessment methodologies.* FT R&D(P4), Peerialism(P6), ZIB(P5), and UCL(P1) completed the definition of a generic, qualitative, and macroscopic methodology for assessing autonomic computing systems. We applied this methodology to the three Selfman application, namely PeerTV from Peerialism(P6), Scalaris from ZIB(P5), and the gPhone application from UCL(P1). We also made progress in the quantitative evaluation of autonomic features of computing systems. We highlight some problems that remain to be solved for doing a generic quantitative benchmark of local autonomic behaviors.

- *Self-configuration services.* We completed the development of a self-configuration framework, including the completion of a third library (WorkflOz) to complement the FructOz and LactOz libraries with the definition of active and self-configurable workflows, and the formal specification of a deployment reference model which provides a language-independent definition of basic functionality required for supporting automated deployment and (re)configuration of component-based distributed systems. The FructOz and LactOz libraries are documented as part of C. Taton's Ph.D. thesis, which was successfully defended on Nov. 2008.

- *Self-healing services.* We completed development of a self-healing framework for distributed clusters. It builds on the Jade autonomous management framework to provide support for the automated recovery from failures in a cluster system, including failures occurring in the management subsystem itself. The work was the subject of the Ph.D. thesis of S. Sicard, which was successfully defended in March 2009.

- *Self-tuning mechanisms.* We designed and evaluated in Scalaris a decentralized algorithm for load balancing on a structured overlay network. The algorithm is fair: the load of individual nodes is close to

the average system load. The algorithm extends Karger's algorithm (in which nodes share load with randomly chosen nodes) with an operation that allows nodes to change position in the overlay, so that the nodes adapt dynamically to better fit the key distribution.

- *Self-protection mechanisms.* We investigated why Small World Networks (SWN) are a better alternative to structured overlay networks (SON) for some security aspects. We showed that SWNs are less efficient thant SONs but still have reasonable performance and can survive node failures like SONs. We showed that real-world social networks have properties similar to SWNs and have natural resilience to Sybil attacks. We developed several defenses against the poisoning attack for SWNs, which can seriously affect the node reorganization algorithm used to get efficient routing. In addition to the work on SWNs, we developed a credibility infrastructure for Wikipedia which is suitable for the open and anonymous nature of Wikipedia.

- *Methodology for building self-managing applications.* Based on the experience in SELFMAN and aided by collaboration with GRID4ALL, we completed a first methodology for building self-managing applications. The methodology covers all four primary self-* properties and their interactions and lays the groundwork for future work in self management.

2.2 Other results

In addition to these major results, we have a series of other results that fit into our overall goal of building self-managing applications.

- KTH(P2) finalized the work on handling network partitioning and mergers for structured overlays.

- Peerialism(P6) developed a tool, NATcracker, to deal with Network Address Translators and Firewalls.

- KTH(P2) and ZIB(P5) optimized the Scalaris system by reducing the validation phase of the commit protocol from 6 to 4 communication steps.

- INRIA(P3) worked on formalizing in Alloy its deployment model, which is independent of the programming language.

- INRIA(P3) started the development of a framework for deploying and configuring applications in a large-scale WAN in the presence of churn.
- UCL(P1) built a transactional store, called Beernet, that extends the Paxos-based transaction algorithm used in Scalaris with notifications and early commit. Beernet is implemented using the Mozart Programming System.
- UCL(P1) released version 1.4.0 of the Mozart Programming System, which provides an advanced network-transparent distribution layer that handles failures using fault streams. Mozart was used to build Beernet.
- UCL(P1) built a prototype of a decentralized collaborative drawing tool on a gPhone by porting Mozart and Beernet to the Android operating system.

3 Workpackage progress of the period

This section gives for each workpackage the objectives, the progress made during the period, the (eventual) deviations and corrective actions, the deliverables, and the milestones. The progress is broken down per partner and a brief explanation of each partner's work is given.

3.1 Workpackage 1: Structured overlay network and basic mechanisms

Workpackage 1 was finished at M24 and has achieved its objectives. Minor work in the subject of this workpackage has been done to maintain the results of this workpackage for the other workpackages (e.g., bug fixes and small extensions to the structured overlay networks). This work is counted as part of the other workpackages.

Objectives To design and implement a structured overlay network that provides the basic self management abilities of node failure/removal/addition, and that provides the self-management primitives (detectors and actuators) needed by the service architecture of WP2.

3.2 Workpackage 2: Service architecture and component model

Workpackage 2 was completed at M24, except for one line of ongoing work, on the Kompics P2P framework, that will result in a M40 deliverable (see second project amendment in Section 4.1).

Objectives To design and implement a distributed component architecture with the basic primitives needed for self management. The component architecture uses the structured overlay network of WP1 as its foundation.

Progress A fairly stable implementation of the SELFMAN architecture and component framework were publicly released in the Kompics open source project (`kompics.sics.se`).

We developed a novel type system for preventing the occurrence of certain configuration errors in component-based communication systems. The work has been applied to two communication frameworks (Dream, developed at INRIA, and Click, developed at MIT). The type system should readily apply to the checking of Kompics v2 configurations.

Deviations and corrective actions None.

Deliverables No deliverables at M36.

Milestones No milestones at M36.

3.3 Workpackage 3: Self-managing storage and transactions

Objectives To design and build a self-managing storage service that provides data replication and the ability to perform transactions. This service will be built on top of the structured overlay network of WP1 and using the component model of WP2. This service is the foundation of the multi-tier application of WP5.

Progress KTH(P2) and ZIB(P5) have worked on designing a component architecture for the transactional DHT algorithms, using the Kompics component model (D3.1c). A Kompics implementation of the transactional key-value store is currently in progress (the implementation will be completed by the review date after M40). UCL(P1) has extended the transaction algorithm

with early commit and notifications. UCL(P1) and ZIB(P5) have worked on visualization tools for the transaction algorithm. ZIB(P5) has completed an open-source release of Scalaris.

Deviations and corrective actions None.

Deliverables

- **D3.1c** (M30): Final report on formal models for transactions over structured overlay networks.
- **D3.2b**: Report on replicated storage service over a structured overlay network.
- **D3.3b**: Report on simple database query layer for replicated storage service.

Milestones No milestones at M36.

3.4 Workpackage 4: Self-management services

Objectives To design and implement the self-management services needed by multi-tier applications. This includes service configuration, reconfiguration, deployment, upgrading during execution, and so forth. We will formalize the self-management services and investigate under what conditions their behavior is convergent.

Progress The work concentrated on the four self-* services. We briefly summarize the work for the four categories, self configuration, self healing, self tuning, and self protection.

Basic mechanisms for self configuration (D4.1b) and self healing (D4.2b) were implemented in the Kompics public release. The work on self healing of overlay networks (in the face of network partitions) has been completed by designing efficient algorithms for the merging of partitioned overlay networks. An important component in managing peer-to-peer networks is to handle NAT traversal. KTH(P2) together with Peerialism(P6) classified all types of NATs and specified the major methods for the discovery of NAT properties and their traversal. Peerialism(P6) developed a technique for NAT traversal which is being used as part of their live-streaming platform PeerTV. We also have a preliminary methodology for using different types of feedback loops in self managing of distributed systems.

We also worked on self-configurable workflows and complemented the FructOz and LactOz libraries with a third library, WorkflOz, which allows the construction of self-configurable workflows. This library can be used on its own to organize an application as a set of distributed cooperating activities, and in conjunction with FructOz and LactOz to build component-based applications with complex deployment and (re)configuration capabilities.

For self healing, we completed the development of a software framework for cluster systems. This framework builds on the Jade management framework to provide support for automatic recovery of failures in a cluster system, including failures occurring in the management subsystem itself. The self-healing framework has been successfully applied to legacy systems, such as a J2EE application server cluster and an NFS file server cluster. The self-healing framework is documented in D4.2b and is presented in detail in S. Sicard's Ph.D. thesis, defended in March 2009.

For self tuning (load balancing), we have developed two algorithms: using workload and key distribution aware placement of nodes and using global knowledge to reduce bandwidth consumption.

For self protection, we developed self-protection mechanisms that focus on Small-World Networks, in the context of studying how to improve the network topology for security. We developed some self-protection mechanisms to help protect SWN mechanisms against some poisoning attacks. We also extended the binary authentication protection mechanism in the second year to extend to application-specific files which could be exploited by an attack.

In addition, we made the following progress:

- We formally specified in Alloy the deployment reference model informally sketched in Year 2. This provides a programming language-independent specification of the basic functionality required to support dynamic deployment and code upgrade in a component-based distributed system. The reference model formalizes the relations between executing components and software deployment units which must be explicitly maintained to ensure cosistent distributed deployment and code upgrade.

- We investigated the problem of deploying and configuring applications in a large-scale WAN environment in presence of churn. In particular, we initiated the development of a framework and protocol for the dynamic slicing of resources in an overlay network, which includes a notion of node profile to take into account the resource requirements of applications and the volatility of nodes. To set up an infrastructure for dynamic slicing of resources in a real-world WAN environment, we

have investigated the question of building a NAT-resilient gossip-based peer-sampling service. Part of the self-deployment and configuration framework is described in C. Taton's Ph.D. thesis, defended in Nov. 2008. The formal specification of the deployment reference model and the WorkflOz library are described in deliverable D4.1b.

Deviations and corrective actions Progress on self configuration in large-scale WAN systems at INRIA was slower than expected. The problem is more complicated than we originally thought. It now appears published dynamic slicing protocols can not be readily exploited for our purposes and that a combination of protocols are required, including a protocol for accountability. No corrective action has been taken except acknowledging the difficulty.

Deliverables

- **D4.1b**: Second report on self-configuration support.
- **D4.1c**: Self-configuration support (software).
- **D4.2b**: Second report on self-healing support.
- **D4.2c**: Self-healing support (software).
- **D4.3b**: Second report on self-tuning support.
- **D4.3c**: Self-tuning support (software).
- **D4.4b**: Self-protection support (software).

Milestones

- **M36**: Finished self-* services on architectural framework.

3.5 Workpackage 5: Application requirements and evaluations

Objectives To build a two-tier application using the service architecture of WP2 and the storage service of WP3, self-managed using the services of WP4. To evaluate and compare standard and self-managing versions of the application. To evaluate and compare the J2EE and Mozart implementations of the application.

Progress Led by FT R&D(P4), we defined the evaluation framework (methodology and process) and then we applied it to the SELFMAN applications. This resulted in the deliverables relative to the qualitative (D5.4a) and quantitative (D5.4b) assessment of self-managing (autonomic) features.

We developed a self-protection infrastructure that gives enhanced security for Wikipedia (explained in D5.6). One part is a credibility mechanism for Wikipedia and WikiMedia which employs the infrastructure together with a MediaWiki extension.

We worked on the MyP2PWorld simulation framework to improve the network emulation layer by adding support for topologies, routing algorithms, and NAT techniques.

We ported Mozart and Beernet to the Android operating system of the HTC Magic gPhone. This port was not trivial because of the primitive C++ support in Android (Mozart is written in C++) and because we needed to create a new graphics binding (Android does not support tcl/tk, Mozart's graphics subsystem). We then modified the TransDraw application to use Beernet and to run on the ported Beernet.

Deviations and corrective actions The two applications developed are a distributed Wikipedia-clone based on Scalaris (D5.2b) and a community-driven recommendation system for Beernet (D5.3). Deliverable D5.3 was developed solely by UCL(P1) with no contribution from INRIA(P3). INRIA(P3) focused on WP4 in Year 3 and had reduced participation in WP5. Deliverable D5.6 does not have any security evaluation for Peerialism(P6), for three reasons: (1) No person-months were allocated for Peerialism(P6) for this work, (2) Peerialism joined the project late and concentrated its efforts on NATs, Kompics, MyP2PWorld, and the autonomic evaluations, and (3) Peerialism has been preoccupied by the recent acquisition procedure initiated by GGF (see Section 4.2.1).

Deliverables

- **D5.2b**: Demonstrator application for J2EE (software).
- **D5.3**: Demonstrator application for Mozart (software).
- **D5.4a**: Qualitative evaluation of self-management properties.
- **D5.4b**: Quantitative evaluation of self-management properties.
- **D5.6**: Evaluation of security mechanisms.
- **D5.7**: Guidelines for developing self-managing applications.

- **D5.8**: Self-managing distributed theater application on mobile devices.

Milestones

- **M36**: Finished dynamic WWW server application (note: this concerns the two demonstrator applications).
- **M36**: Understand effectiveness of self-* services for the application.

3.6 Workpackage 6: Management, dissemination, and exploitation

Objectives To manage the project scientifically and administratively. To maximize the scientific progress. To disseminate the results, including as Open Source software. To collaborate with other projects.

Progress We organized the second project workshop (see Section 5.1.2) that will be held in conjunction with SASO on Sep. 15, 2009. We made several advertisements of the projects in different media (see Section 5.2). We gave invited talks and published the project results as papers and software (see Section 6). We made two project amendments (see Section 4.1): first, we added a demonstrator application for mobile devices (gPhones) and second, we made a project extension for four months.

Deviations and corrective actions The current workshop is the second, not third, because no workshop was held in the first year. Deliverable D6.3 is given as part of this Periodic Activity Report (Sections 5 and 6).

Deliverables

- **D6.1d**: Third project workshop (actually the second).
- **D6.3**: Dissemination and use report.
- **D6.5c**: Final progress and assessment report with lessons learned.

Milestones

- **M36**: Understand general principles of self management.
- **M36**: Underswtand effectiveness of two platform implementation.

4 Consortium management

There were no major problems in the project management during Year 3. We give a summary of the project status, its management and follow-up activities.

4.1 Amendments to the description of work

In this period we have made two amendments to the Description of Work: the first to add a demonstrator application for mobile devices and the second to extend the project for four months.

4.1.1 Amendment of Feb. 11, 2009: new demonstrator application

This amendment adds a new demonstrator application, a distributed collaborative work application for mobile devices, using SELFMAN's transactional storage built on top of a structured overlay network. The budget and person-months per partner are unchanged with this amendment. The application will use Beernet, the Mozart implementation of the scalable storage service of WP3. We will port Mozart to the Android operating system of the HTC Magic mobile phone (gPhone). We will use up to 10 gPhones from the UCL(P1) budget to develop and test the application. We will extend the transactional protocol of Beernet to add notifications and early commit, two properties which are needed for the mobile application.

- T5.8 (new task, UCL(P1) 3 person-months): This task will build an application for mobile devices (specifically, gPhones and laptop computers) that demonstrates the self-managing transactional storage of WP3.
- D5.8 (new deliverable for M36): Self-managing distributed theatre application on mobile devices.

4.1.2 Amendment of Apr. 10, 2009: four month project extension

This amendment extends the project for four months, to a total of 40 months. The budget per partner is unchanged and the total person-months is increased from 317 to 328 (an increase of 3.5%). The reason for this amendment is based on the observation that the project is making good progress and has a small budget surplus at month 36. We therefore decided to request a project extension, to provide extended versions of some of the deliverables,

to allow traveling to the project workshop in September, and to provide a bridge to an eventual follow-up project. This amendment adds the following deliverables due at month 40:

- D2.4: Simulation and emulation environment for Kompics P2P framework (KTH(P2)). This deliverable is part of task T2.2 and extends D2.1c.
- D3.4: Optimizations for self-managing global storage services (ZIB(P5)). This deliverable is part of task T3.2 and extends D3.2b.
- D4.5: Third report on self-configuration support (INRIA(P3)). This deliverable is part of task T4.1 and extends D4.1b.
- D5.9: Distributed mobile application on gPhone (UCL(P1)). This deliverable is part of task T5.8 and extends D5.8.
- D5.10: Design and analysis of Beernet, the Mozart structured overlay network implementation (UCL). This deliverable is part of task T5.3 and consists of the Ph.D. dissertation of Boris Mejias.
- D5.11: Self-protection mechanisms which provide spam resistance (NUS(P7)). This deliverable is part of task T5.6 and extends D5.6.
- D6.1d: Final project workshop. This deliverable consists of the final workshop (second, because there was no workshop the first year) which will be held in conjunction with SASO 2009 on Sep. 15, 2009. See Section 5.1.2 for more information on this workshop.

Except for D5.10, these deliverables are extensions of month 36 deliverables.

4.2 Consortium update

The only relevant change in the consortium during Year 3 concerns Peerialism (see next section).

4.2.1 Peerialism

Peerialism and The Pirate Bay are in process of acquisition by software company Global Gaming Factory X (GGF). The acquisition is intended to be complete by Aug. 2009 if all conditions can be met. The following text is from MarketWatch, Jun. 30, 2009 (`www.marketwatch.com`):

Following the completion of the acquisitions, GGF intends to launch new business models that allow compensation to the content providers and copyright owners. The responsibility for, and operation of the site will be taken over by GGF in connection with closing of the transaction, which is scheduled for August 2009.

GGF has entered into an agreement to acquire the shares in Peerialism AB. Peerialism AB is a software technology company with its origin in KTH Royal Institute of Technology and SICS, Swedish Institute of Computer Science and which presently is owned by the employees. The owners as well as the employees will continue to work for the company. Peerialism develops solutions for data distribution and distributed storage based on new p2p-technology. The access to the technology is secured by the acquisition. The consideration amounts to in aggregate MSEK 100.

"Peerialism has developed a new data distribution technology which now can be introduced on the best known file - sharing site, The Pirate Bay. Since the technology is compatible with the existing it will quickly allow for new values to be created for all key stakeholders and facilitate new business opportunities," says Johan Ljungberg, CEO Peerialism.

"As a result of the acquisitions of The Pirate Bay and Peerialism, GGF will have a strategic position in the international digital distribution market. File sharing traffic is estimated to account for more than half of today's global Internet traffic. The Pirate Bay has a global brand and holds a key position with over 20 million visitors and over one billion page views per month," says Hans Pandeya, [CEO GGF].

4.3 Coordination activities

- Alexander Reinefeld attended the Internet of Services Collaboration meeting for FP6 and FP7 projects, Brussels, Sep. 22-23, 2008.
- Alexander Reinefeld and Seif Haridi attended The European Future Technologies Conference (FET09), Prague, Apr. 21-23, 2009 (see `ec.europa.eu/fet09`).
- Seif Haridi was a member of the Future Internet Services panel at The Future of Internet Conference (FIA09), Prague, May 11-13, 2009.

4.4 Project meetings

We organized the following project meetings during the second year of the project.

- SELFMAN review, Louvain-la-Neuve and Brussels, Belgium (Oct. 2-3, 2008).
- Paris meeting (Dec. 3-4, 2008). Definition of qualitative and quantitative evaluation methodology for autonomic behaviors and the evaluation process for applying this methodology to the SELFMAN applications. Participants from FT R&D(P4) and Peerialism(P6).
- Berlin meeting (Dec. 15-16, 2008). Discussion of application level security issues. Participants from KTH(P2) (representing Peerialism(P6)), NUS(P7), ZIB(P5).
- Virtual meeting (conference call) (Feb. 26, 2009). This meeting was organized by FT R&D(P4) for a presentation and discussion of the self-management assessment methodology. Definition of a schedule: Mar. 15 (listing of all local autonomic behaviors), Mar. 31 (qualitative evaluation of each local autonomic behavior), Apr. 30 (quantitative evaluation of each local autonomic behavior). Participants from Peerialism(P6), ZIB(P5), UCL(P1), NUS(P7), FT R&D(P4).
- Stockholm meeting (April 28-29, 2009). This major meeting was divided into two submeetings. First submeeting: discussion of deliverables, technical talks, and brainstorming on successor project. Second submeeting: discussion to correct problems of evaluating the autonomic behaviors and definition of new planning (May 31 for quantitative evaluation). Participants: KTH(P2), Peerialism(P6), INRIA(P3), FT R&D(P4), UCL(P1), ZIB(P5), NUS(P7). KTH(P2) attendance: Seif Haridi, Cosmin Arad, Tallat Shafaat, Fatemeh Rahimian, Amir Payberah, Ali Ghodsi. INRIA(P3) attendance: Vivien Quema, Jean-Bernard Stefani. FT R&D(P4) attendance: Xavier Etchevers. UCL(P1) attendance: Peter Van Roy, Boris Mejias, Yves Jaradin, Jérémie Melchior. Peerialism(P6) attendance: Roberto Roverso. NUS(P7) attendance: Roland Yap.

4.5 Partner visits

- Mikael Högqvist (ZIB(P5)) made several visits to KTH(P2) and one visit to UCL(P1).

- John Ardelius (SICS) was invited to UCL(P1) to work with Boris Mejías on analyzing the relaxed ring structure.
- Boris Mejías made one visit to SICS to work with John Ardelius.
- Seif Haridi (KTH(P2)) made several visits to ZIB(P5) for cooperation on Scalaris related research.
- Cosmin Arad (KTH(P2)) visited INRIA(P3) (June 25-27, 2008).
- Roland Yap (NUS(P7)) visited ZIB(P5) to work on security issues.

5 Plan for using and disseminating the knowledge

In Year 3 we organized a project workshop, placed two new advertisements (a two-page spread and a Web advertisement), collaborated with three European projects, and made many invited presentations on SELFMAN results. Results are presented on the SELFMAN Wiki at `www.ist-selfman.org` (log on with user `selfman` and password `selfman`).

5.1 Project workshops

5.1.1 First workshop (Oct. 21, 2008)

The first SELFMAN workshop, *Decentralized Self Management for Grids, P2P, and User Communities*, was held in conjunction with SASO 2008 in Venice, Italy, Oct. 20-24, 2008:

`www.ist-selfman.org/wiki/index.php/SelfmanWorkshop`

The workshop was co-sponsored by the SELFMAN and GRID4ALL projects. The organizing committee of this workship consisted of Peter Van Roy, Marc Shapiro, and Seif Haridi. The program committee consisted of Gustavo Alonso, Seif Haridi, Bernardo Huberman, Adriana Iamnitchi, Mark S. Miller, Pascal Molli, Luc Onana Alima, Nuno Preguiça, Alexander Reinefeld, Marc Shapiro, Peter Van Roy, and Hakim Weatherspoon. We received 18 submissions, each of which was reviewed by three program committee members, and we accepted 15 papers. The electronic proceedings of this workshop, 104 pages long, was published by IEEE.

5.1.2 Second workshop (Sep. 15, 2009)

The second SELFMAN workshop, *Architectures and Languages for Self-Managing Distributed Systems*, will be held in conjunction with SASO 2009 in San Francisco, CA, Sep. 14-18, 2009:

`www.ist-selfman.org/wiki/index.php/SelfmanWorkshop2009`

Important dates for the workshop are:

- Submission of position paper: July 10, 2009 (required for attendance)
- Notification of acceptance: August 5, 2009
- Final copy: August 19, 2009
- Workshop: September 15, 2009

The workshop is sponsored by the SELFMAN project. The organizing committee consists of Jean-Bernard Stefani, Seif Haridi, and Peter Van Roy. The keynote speaker will be Terence Kelly, HP Labs, Palo Alto, CA. The program committee consists of:

- Gordon Blair, Lancaster University, UK
- Pierre Cointe, École des Mines, Nantes, France
- Thierry Coupaye, Orange Labs, France
- Jean-Charles Fabre, Institut National Polytechnique, Toulouse, France
- Seif Haridi, SICS & KTH, Stockholm, Sweden
- Tom Holvoet, Katholieke Universiteit Leuven, Belgium
- Mark Jelasity, University of Szeged, Hungary
- Emre Kiciman, Microsoft Research, Redmond, WA, USA
- Mark S. Miller, Google Research, USA
- Alexander Reinefeld, Zuse Institute, Berlin, Germany
- Bradley Schmerl, Carnegie-Mellon University, Pittsburgh, PA, USA
- Jean-Bernard Stefani, INRIA, Grenoble, France
- Alexander Wolf, Imperial College, London, UK

- Peter Van Roy, Université catholique de Louvain, Belgium

The goal of the workshop is to bring together researchers and practitioners interested in the construction of self-managing distributed systems. It will place the emphasis on software engineering (especially, software architecture and component-based software engineering) aspects of this construction, including models, architectures, languages, control techniques, middleware and tools that can be used to support the modular and principled building of self-* distributed systems. Topics of interest for the workshop include (but are not limited to):

- Component models and architectures for self management.
- Generative and reflective (including aspect-oriented) techniques for self management.
- Languages for self-managing systems, including formal specification, architecture description, programming, and domain specific languages.
- Control techniques for self-managing systems, including control-theoretic and decision-theoretic techniques.
- Coordination and decentralized architectures of control.
- Analysis and verification techniques for self-managing systems.
- Middleware and tool support for self-managing distributed systems.
- Algorithms for distributed self management, including event detection, distributed control, etc.

Application areas of interest to the workshop include (but are not limited to): Web services, social networks, cloud computing, P2P systems and applications, pervasive computing.

5.2 Advertisements

In Year 3, four advertisements appeared for SELFMAN and its results:

- A four-page article, *A Scalable, Transactional Data Store for Web 2.0 Services*. This article is included at the end of this Activity Report. An extended version of this article was published as a book chapter (see Section 6.4) and is included in the appendix of the collected deliverables for Year 3.

PeerTV:
The Future of Media Distribution

PeerTV is a comprehensive application for end users, content owners, and broadband operators that distributes video over Internet with live streaming and progressive download. It was recently launched in the Swedish market by startup company Peerialism. It offers its customers lower cost of distribution, reduced peak capacity investments, as well as higher viewing capacity (more simultaneous viewers), through advanced technology such as traffic optimization and P2P overlay networks.

To reach the objectives for PeerTV and to meet other customer requirements, Peerialism has developed advanced technology in several areas: innovative P2P components including P2P protocols and optimization algorithms (such as Chandelier) targeted towards live streaming, an integrated simulation and emulation tool (MyP2PWorld) to radically speed up development and shorten time to market, and new techniques for firewall hole punching and NAT traversal of video streams. The result is a new, highly capable video distribution solution built on structured P2P overlays developed from scratch in little over 12 months. PeerTV has comparable QoS to leading distribution providers but at much lower costs. PeerTV is ISP friendly as well: traffic is to a large degree kept local in the network, and is transparent and encoding agnostic. The distribution layer is transparent to media servers and media players.

A key success factor in the current and future development work is direct access to advanced research in distributed systems. Peerialism has close ties to the Swedish Institute of Computer Science (SICS) and the Royal Institute of Technology (KTH) and is a partner in several Swedish and European projects including the SELFMAN project. The connections with institutes like SICS and projects like SELFMAN allow the company to quickly test and evaluate new P2P components. The company receives valuable feedback on its own work and is provided access to work by others; for instance, the company is currently testing the SELFMAN component model Kompics (KTH) and is investigating potential benefits of closer integration with the load injection framework CLIF (France Telecom), and the scalable transactional store Scalaris (ZIB).

Peerialism currently employs 14 people with offices in Stockholm and Cairo. It finances its commercialization and growth out of its own revenues. An internal program to allow employees to do a PhD whilst being employed by the company has recently been launched.

For additional information please see **www.peerialism.com** .

PEERIALISM

SELFMAN

Building large-scale self-managing distributed systems based on structured P2P overlay networks and software components.

European sixth framework programme project, IST Research in Software Technologies, June 2006–May 2009, 1.96M€ budget.

Peter Van Roy
(project coordinator)
Université catholique de Louvain
Email: peter.vanroy@uclouvain.be
Tel: +32 485 42 46 77
Web: www.ist-selfman.org

THE ADVENTURES OF SELFMAN

DISSEMINATION

Figure 1: Advertisement for PeerTV in Research Review

Scalaris: Scalable Transactional Storage for Web 2.0 Services

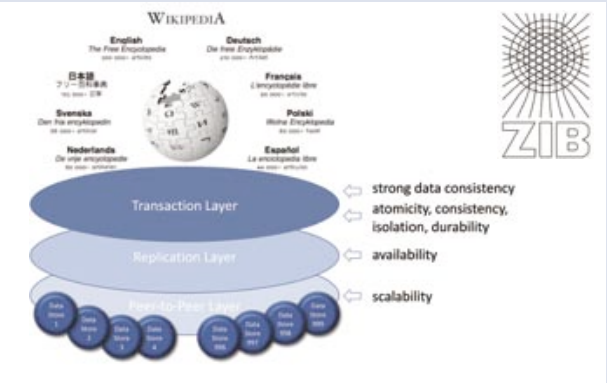
Scalaris provides a self-managing scalable, transactional storage for large-scale distributed Web 2.0 services. It provides a distributed key/value store built on top of a replicated storage layer and an enhanced structured P2P overlay network. Scalaris provides highly available transactional support for strong data consistency in the face of concurrent data operations, computing node failures, and network problems. It uses a fast consensus protocol with low communication overhead that has been optimally embedded into the P2P overlay network. Scalaris is an important building block for the core of Web 2.0 applications and future Cloud Computing environments. Scalaris was implemented by the Zuse Institute Berlin (ZIB) as part of the European SELFMAN project.

We demonstrated the capabilities of Scalaris by reimplementing the core of Wikipedia. This implementation won first prize in the IEEE International Scalable Computing Challenge 2008. The implementation is both fast and scalable: using eight servers it executes 2,500 transactions per second. All operations are performed within transactions to guarantee data consistency and replica synchronization. Adding more computers improves the performance almost linearly. The public Wikipedia, in contrast, employs ten servers to execute the 2,000 requests per second that hit the backend on its large master/slave MySQL database (Wikipedia handles around 50,000 requests per second, with 48,000 handled by proxies).

For many Web 2.0 services, the total cost-of-ownership is dominated by the costs needed for personnel to maintain and optimize the service. Scalaris greatly reduces these costs with its built-in self-management properties. It is self healing: when it detects a computing node crash or network problem, it immediately repairs its P2P overlay network and the database. Management tasks such as adding or removing nodes require no or very little human intervention. It is self tuning: it autonomously moves items to distribute the load evenly over the system to improve the response time. When deploying Scalaris over multiple data centers, these algorithms are used to place frequently accessed items near the users.

The Zuse Institute Berlin is a research institute for applied mathematics and computer science located in Berlin. The Scalaris source code is released under an open source license and is available at code.google.com/p/scalaris/.

For additional information please see **www.zib.de** and **www.onscale.de** .



SELFMAN

Building large-scale self-managing distributed systems based on structured P2P overlay networks and software components.

European sixth framework programme project, IST Research in Software Technologies, June 2006–May 2009, 1.96M€ budget.

Peter Van Roy
(project coordinator)
Université catholique de Louvain
Email: peter.vanroy@uclouvain.be
Tel: +32 485 42 46 77
Web: www.ist-selfman.org

THE ADVENTURES OF SELFMAN

DISSEMINATION

Figure 2: Advertisement for Scalaris in Research Review

- A two-page spread in *Research Review* magazine (Issue 7, Nov. 2008, pages 26-27). The left page advertises the PeerTV product (see Figure 1) and the right page advertises the Scalaris library (see Figure 2). We retained the copyright to this two-page spread and we have distributed it to interested parties.

- A Web advertisement that appeared on the Web portal of PSCA International Ltd. (`www.publicservice.co.uk`) for one year (Dec. 2008 - Dec. 2009). To see the advertisement, click first the PUBLICATIONS tab (on top) and then the European Union label (at the left). Then the advertisement appears in the rightmost column as a clickable box.

- A three-page dissemination article, *A Self-Managing Peer-to-Peer Network*, which appeared in the magazine *eStrategies Projects*, published by British Publishers in June 2008. See the Year 2 Activity Report for the full text.

5.3 Collaboration

- *XtreemOS* project (`www.xtreemos.org`). In this collaboration, a part of the scalable Scalaris key/value store (namely the DHT and the transaction framework) was used to implement a scalable metadata store and a publish/subscribe system. As a result of this collaboration Mandriva is now offering Scalaris RPMs.

- *GRID4ALL* project (`www.grid4all.eu`). In this collaboration (with KTH and Orange Labs) we worked on automated deployment and deployment on overlay networks. We worked with KTH on developing a framework for self-managing component-based applications. The framework extends the Fractal component model by the component group abstraction and one-to-any and one-to-all bindings between components and groups. The framework supports a network-transparent view of system architecture simplifying designing application self-managing code. We exploited ideas from the SELFMAN work on self configuration to develop the deployment service included in this framework. We also collaborated on an initial methodology for using different patterns of feedback loops in system design. This led to the notions of decomposition and orchestration in deliverable D5.7 (the Guidelines).

- *MANCOOSI* project (`www.mancoosi.org`). MANCOOSI is working on package management for open-source installations. We have an ongo-

ing discussion on how to use the distributed technology of SELFMAN for two purposes: coherent distributed installation with rollback and distributed information sharing to improve the efficiency of package installability solving, which is highly compute-intensive.

5.4 Invited talks and similar activities

- Joe Armstrong. The developer of Erlang, Joe Armstrong, posted an article about Scalaris on his blog (Jun. 28, 2008). It is available here:

  `armstrongonsoftware.blogspot.com/2008/06/itching-my-programming-nerve.html`

- Peter Van Roy. *The Challenges and Opportunities of Multiple Processors: Why Multi-Core Processors are Easy and Internet is Hard*, position statement, panel discussion on *Reinventing Audio and Music Computation for Many-Core Processors*, International Computer Music Conference (ICMC 2008), Belfast, Ireland, Aug. 24-29, 2008.

- Thorsten Schütt and Alexander Reinefeld. Invited talk on Scalaris, Google Zurich, Sep. 11, 2008.

- Jean-Bernard Stefani. Participant at the IST Coordination Meeting, Brussels, Belgium, Sep. 22-23, 2008.

- Thorsten Schütt. Invited talk on Scalaris, Humboldt University, Oct. 10, 2008.

- Seif Haridi. Member of the panel session, Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008), Venice, Italy, Oct. 20-24, 2008.

- Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. *A Scalable, Transactional Data Store for Web 2.0 Services: Implementing Wikipedia with Scalaris*, article that appeared on `TheServerSide.com`, Nov. 2008. This article is included at the end of this Activity Report.

- Thorsten Schütt. Invited talk on Scalaris, University Kaiserslautern, Nov. 26, 2008.

- Boris Mejías. *Self-Managing Large-Scale Decentralized Systems*, presentation at doctoral symposium, EuroDocInfo 09, University of Mons, Mons, Belgium, Jan. 22-23, 2009.

- Florian Schintke and Alexander Reinefeld. *Implementing Fault Tolerant Services on Structured Overlay Networks: Parts 1 and 2 of the Messy Details* (two talks), Dagstuhl Workshop on Fault Tolerance in High-Performance Computing and Grids, May 4, 2009.
- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Network Size Estimation in Structured Overlay Networks*, Sixth Swedish National Computer Networking Workshop (SNCNW '09), Uppsala, Sweden, May 4-5, 2009.
- Alexander Reinefeld. Invited talk on Scalaris, European Media Laboratory GmbH (EML), Heidelberg, May 9, 2009.
- Seif Haridi. Member of the panel on Future Internet Services, EU Future of the Internet Conference, Prague, May 11-13, 2009.
- Alexander Reinefeld, Florian Schintke, Thorsten Schütt, and Seif Haridi. *A Scalable, Transactional Data Store for Future Internet Services*, EU Future of the Internet, Prague, May 11-13, 2009.
- Boris Mejías. *Self Management of Large-Scale Distributed Systems*, invited talk, Universiteit Antwerpen, Antwerpen, Belgium, May 18, 2009.
- Boris Mejías. *Self Management of Large-Scale Distributed Systems*, presentation at Dept. of Computing Science and Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Apr. 21, 2009.
- Florian Schintke. *Scalaris: Methods for a Globally Distributed Key-Value Store*, Data-Aware Distributed Computing Workshop (DADC 2009), Munich, Jun. 9-13, 2009.

6 Papers and software

This section lists the papers and other publications (including software) that were funded in whole or part by SELFMAN.

6.1 Software releases

- *SicSim* discrete event simulator for simulating peer-to-peer overlays. Public release by KTH(P2) as free software under the GNU GPL version 2 on Dec. 2008. Available at `www.sics.se/~amir/sicsim`. (Note: this simulator was first called SicsSim; some articles in the deliverables still have this old name.)

- *Kompics* Reactive Component Model for Distributed Computing. Public release by KTH(P2) as free software under the GNU GPL version 2 on Feb. 4, 2009. Available at `kompics.sics.se`.
- *Scalaris* transactional key-value store. Public release by ZIB(P5) as open-source software under the Apache License on May 20, 2009. Available at `scalaris.googlecode.com`.
- *Mozart Programming System* version 1.4.0. Public release by UCL(P1) as open-source software (X11 style license) on July 3, 2008. Available at `www.mozart-oz.org`.

Software still under development

- *MyP2PWorld* application-level network and concurrency emulator. To be released by Peerialism(P6).
- *Beernet* transactional key-value store. To be released by UCL(P1) as open-source software (X11 style license). Available at `p2ps.info.ucl.ac.be`. This software was developed in Mozart 1.3.2 and subsequently redesigned to take advantage of the possibilities of 1.4.0.
- *WorkflOz/FructOz/LactOz* configuration management libraries. To be released by INRIA(P3) as free software (LGPL license). Available upon request. This software was developed in Mozart 1.3.2.

6.2 Journal papers

- Boris Mejías and Peter Van Roy. *The Relaxed-Ring: A Fault-Tolerant Topology for Structured Overlay Networks*, Journal of Parallel Processing Letters, Vol. 18(3):411–432, World Scientific, Sep. 2008.
- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Dealing with Network Partitions in Structured Overlay Networks*, Journal of Peer-to-Peer Networking and Applications (PPNA), Springer (to appear).

6.3 Dissertations and licentiate theses

- Christophe Taton. "Vers l'Auto-Optimisation dans les Systèmes Autonomes" (Towards Self-Optimization in Autonomic Systems), Ph.D. dissertation, Institut Polytechnique de Grenoble, Grenoble, France, Dec. 2008. Jury: Roger Mohr, Christine Morin, Peter Van Roy, Jacques Mossière, Sara Bouchenak, Marta Pariño-Martínez.

- Felix Hupfeld. "Causal Weak-Consistency Replication: A Systems Approach," Ph. D. dissertation, Humboldt Universität zu Berlin, Germany, Jan. 28, 2009. Reviewers: Alexander Reinefeld, Marc Shapiro, Jens-Peter Redlich.

- Sylvain Sicard. "Vers l'Auto-Réparation dans les Systèmes Répartis" (Towards Self-Healing in Distributed Systems), Ph. D. dissertation, Université Joseph Fourier, Grenoble, France, Mar. 2009. Advisor: Jean-Bernard Stefani.

- Tallat Mahmood Shafaat. "Dealing with Network Partitions and Mergers in Structured Overlay Networks," Licentiate Thesis, KTH School of Information and Communication Technology, Stockholm, Sweden, May 2009.

- Cosmin Arad. "Kompics: A Concurrent Component Model for Distributed Systems," Licentiate Thesis, KTH School of Information and Communication Technology, Stockholm, Sweden, 2009 (in preparation).

6.4 Book chapters

- F. Boyer, N. De Palma, O. Gruber, S. Sicard and J.B. Stefani. *A Self-Repair Architecture for Cluster Systems*, chapter in Architecting Dependable Systems 6, R. de Lemos, J.-C. Fabre, C. Gacek, F. Gadducci, M.H. ter Beek (eds.), Springer, 2009.

- Alexander Reinefeld, Florian Schintke, Thorsten Schütt, and Seif Haridi. *A Scalable, Transactional Data Store for Future Internet Services*, chapter in Towards the Future Internet–A European Research Perspective, G. Tselentis *et al.* (eds.), IOS Press, 2009.

- Peter Van Roy. *Programming Paradigms for Dummies: What Every Programmer Should Know*, chapter in New Computational Paradigms for Computer Music, G. Assayag and A. Gerzso (eds.), IRCAM/Delatour France, May 2009.

- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Managing Network Partitions in Structured P2P Networks*, chapter in Handbook of Peer-to-Peer Networking, X. Shen, H. Yu, J. Buford, M. Akon (eds.), Springer, Dec. 2009 (to appear).

- Yongzheng Wu, Sufatrio, Roland H.C. Yap, R. Ramnath, and Felix Halim, *Establishing Software Integrity Trust: A Survey and Lightweight Authentication System for Windows*, chapter in Trust Modeling and Management in Digital Environments: from Social Concept to System Development, Zheng Yan (ed.), IGI Global, Dec. 2009 (to appear).

6.5 Conference papers

- T. Shafaat, M. Moser, A. Ghodsi, T. Schütt, S. Haridi, and A. Reinefeld. *Key-Based Consistency and Availability in Structured Overlay Networks*, Third International ICST Conference on Scalable Information Systems (Infoscale'08), Vico Equense, Italy, Jun. 4-6, 2008.

- Peter Van Roy. *Overcoming Software Fragility with Interacting Feedback Loops and Reversible Phase Transitions*, First International Conference on Visions of Computer Science (BCS 08), London, UK, Sep. 22-24, 2008.

- Felix Halim, Yongzheng Wu, and Roland H.C. Yap. *Security Issues in Small World Network Routing*, Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008), Venice, Italy, Oct. 20-24, 2008.

- Alfredo Cádiz, Boris Mejías, Jorge Vallejos, Kim Mens, Peter Van Roy, and Wolfgang de Meuter. *PALTA: Peer-to-peer AdaptabLe Topology for Ambient intelligence*, XXVII IEEE International Conference of the Chilean Computer Science Society (SCCC 2008), Punta Arenas, Chile, Nov. 13-14, 2008.

6.6 Future conferences and workshops

Accepted submissions

- Michaël Lienhardt, Claudio Antares Mezzina, Alan Schmitt, and Jean-Bernard Stefani. *Typing Component-Based Communication Systems*, IFIP International Conference on Formal Techniques for Distributed Systems, formed jointly from 11th Formal Methods for Open Object-Based Distributed Systems and 29th Formal Techniques for Networked and Distributed Systems (FMOODS/FORTE), Springer LNCS, Lisbon, Portugal, Jun. 9-12, 2009.

- Cosmin Arad, Jim Dowling, and Seif Haridi. *Developing, Simulating, and Deploying Peer-to-Peer Systems using the Kompics Component*

Model, Fourth International Conference on COMmunication System software and middlewaRE (COMSWARE '09), Dublin, Ireland, Jun. 16-19, 2009.

- Anne-Marie Kermarrec, Alessio Pace, Vivien Quéma, and Valerio Schiavoni. *NAT-resilient Gossip Peer Sampling*, 29th International Conference on Distributed Computing Systems (ICDCS), Montreal, Quebec, Jun. 22-26, 2009.

- Boris Mejías, Alfredi Cádiz, and Peter Van Roy. *Beernet: RMI-Free Peer-to-Peer Networks*, First International Workshop on Distributed Objects for the 21st Century, colocated with ECOOP 2009, Genova, Italy, Jul. 7, 2009.

- Jérémie Melchior, Donatien Grolaux, Jean Vanderdonckt, and Peter Van Roy. *A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications*, ACM SIGCHI Symposium on Engineering Interactive Computer Systems (EICS 2009), Pittsburgh, PA, Jul. 14-17, 2009.

- Roberto Roverso, Sameh El-Ansary, and Seif Haridi. *NATCracker: NAT Combinations Matter*, 18th International Conference on Computer Communications and Networks (ICCCN 2009), San Francisco, CA, Aug. 3-6, 2009.

- Ahmad Al-Shishtawy, Vladimir Vlassov, Per Brand, and Seif Haridi. *A Design Methodology for Self-Management in Distributed Environments*, 2009 IEEE International Symposium on Scientific and Engineering Computing (SEC-09), Vancouver, Canada, Aug. 29-31, 2009.

- T. Schütt, M. Hoffmann, F. Schintke, and A. Reinefeld. *Gossip-based Topology Inference for Efficient Overlay Mapping*, short paper, Ninth International Conference on Peer-to-Peer Computing (P2P 2009), Seattle, WA, Sep. 8-11, 2009.

- Cosmin Arad, Jim Dowling, and Seif Haridi. *Evaluating P2P Systems in the Kompics Component Framework*, invited talk and demonstrator, Ninth International Conference on Peer-to-Peer Computing (P2P 2009), Seattle, WA, Sep. 8-11, 2009.

- Felix Halim, Yongzheng Wu, and Roland H.C. Yap. *Wiki Credibility Enhancement*, Fifth International Symposium on Wikis and Open Collaboration (WikiSym 2009), Orlando, FL, Oct. 25-27, 2009.

Submissions in progress

- Xavier Etchevers and Thierry Coupaye. *Benchmarking Autonomic Systems from a Technical and an Economical Perspective*, Third International ICST Conference on Autonomic Computing and Communication Systems (Autonomics 2009), Limassol, Cyprus, Sep. 9-11, 2009.
- Mikael Högqvist. *Impact of Non-Uniform Key Distribution and Workload in Distributed Key/Value Stores*, 5th International Workshop on Networking Meets Databases (NetDB 2009), colocated with SOSP 2009, Big Sky, MT, Oct. 14, 2009.
- Florian Schintke, Alexander Reinefeld, and Seif Haridi. *Enhanced Paxos Commit for Transactions on DHTs*, 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009), Lyon, France, Nov. 3-6, 2009.

6.7 Workshop papers, demonstrators, and posters

- Paolo Costa, Guillaume Pierre, Alexander Reinefeld, Thorsten Schütt, and Maarten van Steen. *Sloppy Management of Structured P2P Services*, 3rd Workshop on Hot Topics in Autonomic Computing (HotAC), Chicago, IL, Jun. 2, 2008.
- Tallat M. Shafaat, Thorsten Schütt, Monika Moser, Seif Haridi, Ali Ghodsi, and Alexander Reinefeld. *Key-Based Consistency and Availability in Structured Overlay Networks*, poster at 17th International Symposium on High-Performance Distributed Computing (HPDC 2008), Boston, MA, Jun. 23-27, 2008.
- Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. *Scalaris: Reliable Transactional P2P Key/Value Store - Web 2.0 Hosting with Erlang and Java*, 7th ACM SIGPLAN Erlang Workshop, Victoria, BC, Canada, Sep. 2008.
- Boris Mejias, Mikael Högqvist, and Peter Van Roy. *Visualizing Transactional Algorithms for DHTs*, demonstrator at Eighth International Conference on Peer-to-Peer Computing (P2P 2008), Aachen, Germany, Sep. 8-11, 2008.
- Cosmin Arad and Seif Haridi. *Practical Protocol Composition, Encapsulation, and Sharing in Kompics*, IEEE SASO Workshop on Decentralized Self Management for Grid, P2P, and User Communities (SELFMAN workshop), Venice, Italy, Oct. 21, 2008.

- Mikael Högqvist, Seif Haridi, Nico Kruber, Alexander Reinefeld, and Thorsten Schütt, *Using Global Information for Load Balancing in DHTs*, IEEE SASO Workshop on Decentralized Self Management for Grid, P2P, and User Communities (SELFMAN workshop), Venice, Italy, Oct. 21, 2008.

- Boris Mejías, Alfredo Cádiz, Peter Van Roy, and Kim Mens. *A Self-Adaptable Network Topology for Ambient Intelligence*, IEEE SASO Workshop on Decentralized Self Management for Grid, P2P, and User Communities (SELFMAN workshop), Venice, Italy, Oct. 21, 2008.

- Gustavo Gutiérrez, Boris Mejías, Peter Van Roy, Diana Velasco, and Juan Torres. *WSN and P2P: A Self-Managing Marriage*, IEEE SASO Workshop on Decentralized Self Management for Grid, P2P, and User Communities (SELFMAN workshop), Venice, Italy, Oct. 21, 2008.

- Felix Halim, Yongzheng Wu, and Roland H.C. Yap. *Small World Networks as (Semi)-Structured Overlay Networks*, IEEE SASO Workshop on Decentralized Self Management for Grid, P2P, and User Communities (SELFMAN workshop), Venice, Italy, Oct. 21, 2008.

- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *A Practical Approach to Network Size Estimation for Structured Overlays*, Third International Workshop on Self-Organizing Systems (IWSOS 2008), Vienna, Austria, Dec. 10-12, 2008.

- Mikael Högqvist. *Architecture and Self-Tuning of a DISC-system*, poster at TCPP PhD Symposium, IPDPS 2009, Rome, Italy, May 25-29, 2009.

- Boris Mejías, Jérémie Melchior, and Yves Jaradin. *DeTransDraw: Decentralized Transactional Collaborative Drawing*, demonstrator in Internet of Service 2009, ICT Challenge 1.2, Collaboration Meeting for FP6 & FP7 projects, Brussels, Belgium, Jun. 10-11, 2009.

7 Four-page advertisement (see Section 5.2)

A SCALABLE, TRANSACTIONAL DATA STORE FOR WEB 2.0 SERVICES

IMPLEMENTING WIKIPEDIA WITH SCALARIS

Web 2.0 – the Internet as an information society platform supporting business, recreation and knowledge exchange – initiated a business revolution. Service providers offer Internet services for shopping (Amazon, eBay), online banking, information (Google, Flickr, Wikipedia), social networking (MySpace, Facebook), and recreation (Second Life, online games). In our information society, Web 2.0 services are no longer a thing that is just nice to have, but customers today depend on their continuous availability, regardless of time and space.

How to cope with such strong demands, especially in case of interactive community services that cannot be simply replicated? All users access the same Wikipedia, meet in the same Second Life environment and want to discuss with others via Twitter. Even the shortest interruption, caused by system downtime or network partitioning may cause huge losses in reputation and revenue. Web 2.0 services are not just an added value, but they must be dependable. Apart from 24/7 **availability**, providers face another challenge: they must, for a good user experience, be able to respond within milliseconds to incoming requests, regardless whether thousands or millions of concurrent requests are currently being served. Indeed, **scalability** is a key challenge. Any scalable service, to be affordable, somehow requires the system to be self managing (see sidebar).

Availability is the proportion of time a system is in a functioning condition. More formally, availability is a ratio of the expected value of the uptime of a system to the aggregate of the expected values of up and down time. Availability is often specified in a logarithmic unit called “nines”, which corresponds roughly to a number of nines following the decimal point. “Six nines”, for example, denote an availability of 0.999999, allowing a maximum downtime of 31 seconds per year.

Scalability refers to the capability of a system to increase the total throughput under an increased load when resources are added. A scalable database management system is one that can be upgraded to process more transactions by adding new processors, devices and storage, and which can be upgraded easily and transparently without service interrupt.

Self Management refers to the ability of a system to adjust to changing operating conditions and requirements without human intervention at runtime. Self Management includes self configuration, self healing and self tuning.

Our **Scalaris** system, described below, provides a comprehensive solution for self managing, scalable data management. In our opinion, Scalaris and similar systems will be an important core service of future **Cloud Computing** environments.

As a common key aspect, all Web 2.0 services have to deal with concurrent data updates. Typical examples are checking the availability of products and their prices, purchasing items and putting them into virtual shopping carts, and updating the state in multi-player online games. Clearly, many of these data operations have to be atomic, consistent, isolated and durable (so-called ACID properties). Traditional centralized database systems are ill-suited for this task, sooner or later they become a bottleneck for business workflow. Rather, a scalable, transactional data store like Scalaris is what is needed.

SCALARIS KEY/VALUE STORE

As part of the EU funded SELFMAN[1] project we set out to build a distributed key/value store capable of serving thousands or even millions of concurrent data accesses per second. Providing strong data consistency in the face of node crashes and hefty concurrent write accesses was one of our major goals.

With our **Scalaris** system, we do not attempt to replace current database management systems with their general, full-fledged SQL interfaces. Instead our target is to support transactional Web 2.0 services like those needed for Internet shopping, banking, or multi-player online games. Our system consists of three layers:

- At the bottom, an enhanced structured overlay network, with logarithmic routing performance, provides the basis for storing and retrieving keys and their corresponding values. In contrast to many other overlays, our implementation stores the keys in lexicographical order. Lexicographical ordering instead of random hashing enables control of data placement which is necessary for low latency access in multi-datacenter environments.

- The middle layer implements data replication. It enhances the availability of data even under harsh conditions such as node crashes and physical network failures.

[1] SELFMAN is a specific targeted research project funded in the 6th framework programme of the EU under contract no. 34084.

- The top layer provides transactional support for strong data consistency in the face of concurrent data operations. It uses a fast consensus protocol with low communication overhead that has been optimally embedded into the structured overlay.

Together, these three layers provide a distributed key/value store as a scalable and highly available service which is an important building block for Web 2.0 applications.

WIKIPEDIA ON SCALARIS

As a challenging benchmark for Scalaris, we implemented the core of Wikipedia, the "free encyclopedia, that anyone can edit". Wikipedia runs on three sites. The main one in Tampa is organized in three layers, the proxy server layer, the web server layer, and the MySQL database layer. The proxy layer serves as a cache for recent requests, and the web server layer runs the application logic and issues requests to the data base layer. Wikipedia handles about 50,000 requests per second, from which 48,000 are cache hits in the proxy server layer and 2,000 are processed by the data base layer. The proxy and the web server layers are embarrassingly parallel and therefore trivial to scale. From a scalability point of view, only the data base layer is challenging.

Our implementation uses Scalaris to replace the data base layer. This enables us to run Wikipedia on geographically distributed sites and to scale to almost any number of hosts. It inherits all the favorable properties of Scalaris, such as scalability and self management.

The Wikipedia on Scalaris is fast. Using eight servers it executes 2,500 transactions per second. All operations are performed within transactions to guarantee data consistency and replica synchronization. Adding more computers improves the performance almost linearly. The public Wikipedia, in contrast, employs ten servers to execute the 2,000 requests per second on the large master/slave MySQL database in Tampa.

SELF-MANAGEMENT

For many Web 2.0 services, the total cost-of-ownership is dominated by the costs needed for personnel to maintain and optimize the service. Scalaris greatly reduces the operation cost with its built-in self* properties:

- **Self healing**: Scalaris continuously monitors the hosts it is running on. When it detects a node crash, it immediately repairs the overlay network and the database. Management tasks such as adding or removing hosts require minimal human intervention.

- **Self tuning:** Scalaris monitors the nodes' workload and autonomously moves items to distribute the load evenly over the system to improve the response time of the system. When deploying Scalaris over multiple data-centers, these algorithms are used to place frequently accessed items nearby the users.

In traditional database systems these operations require human interference which is error prone and costly. With Scalaris the same number of system administrators can operate much larger installations than with legacy databases.

SUMMARY

Scalaris provides a scalable and self managing transactional key-value store. We have implemented Wikipedia using Scalaris. Its scalability and self* capabilities were demonstrated in the IEEE Scalable Computing Challenge 2008, where Scalaris won the 1st prize (see plaque).

Compared to other data services, Scalaris has significantly lower operating costs. Scalaris and similar systems will be an important building block for Web 2.0 services and future Cloud Computing environments.

ADDITIONAL INFORMATION

- For the EU project Selfman see http://www.ist-selfman.org

- The Scalaris code is open source. It is available at http://code.google.com/p/scalaris/. Additional information (papers, videos) can be found at http://www.zib.de and http://www.onscale.de.

8 Article on TheServerSide.com (see Section 5.4)

By Thorsten Schuett, Florian Schintke, and Alexander Reinefeld

A Scalable, Transactional Data Store for Web 2.0 Services

Implementing Wikipedia With Scalaris

November 2008

Discuss this Article

Global online services such as Amazon, eBay, Myspace, YouTube, and Google serve millions of customers through tens of thousands of servers located world-wide. On this immense scale, components fail continuously and it is very difficult to maintain a consistent state while at the same time hiding failures from the application.

Peer-to-peer protocols achieve self-management by replicating services among peers, but these are mostly limited to write-once/read-many data sharing. To extend them beyond typical file sharing, the support of fast transactions on distributed hash tables (DHTs) is an important, and until now, elusive piece of functionality.

The Scalaris system, described below, provides a comprehensive solution for self managing, scalable data management. Scalaris and similar systems may be an important core service of future cloud computing environments.

As a common key aspect, all Web 2.0 services have to deal with concurrent data updates. Typical examples are checking the availability of products and their prices, purchasing items and putting them into virtual shopping carts, and updating the state in multi-player online games. Clearly, many of these data operations have to be atomic, consistent, isolated and durable (ACID). Traditional centralized database systems are ill-suited for this task, sooner or later they become a bottleneck for business workflow. Rather, a scalable, transactional data store like Scalaris is what is needed.

Scalaris Key/Value Store

We set out to build a distributed key/value store capable of serving thousands or even millions of concurrent data accesses per second. Providing strong data consistency in the face of node crashes and hefty concurrent write accesses was one of our major goals.

With the Scalaris system, we do not attempt to replace current database management systems with their general, full-fledged SQL interfaces. Instead our target is to support transactional Web 2.0 services like those needed for Internet shopping, banking, or multi-player online games. Our system consists of three layers:

- At the bottom, an enhanced structured overlay network, with logarithmic routing performance, provides the basis for storing and retrieving keys and their corresponding values. In contrast to many other overlays, our implementation stores the keys in lexicographical order. Lexicographical ordering instead of random hashing enables control of data placement which is necessary for low latency access in multi-datacenter environments.
- The middle layer implements data replication. It enhances the availability of data even under harsh conditions such as node crashes and physical network failures.
- The top layer provides transactional support for strong data consistency in the face of concurrent data operations. It uses a fast consensus protocol with low communication overhead that has been optimally embedded into the structured overlay.

These three layers are all implemented in Erlang. Together, they provide a distributed key/value store as a scalable and highly available service which is an important building block for Web 2.0 applications.

Why use Erlang?

Erlang is a functional programming language. It provides only write-once variables, and is therefore often cited as the ideal programming language for implementing parallel algorithms. However, Erlang is more than simply a great programming language – it also makes it very easy to convert parallel programs into distributed ones, because the concurrency model is based on message passing instead of shared state. By distributing the individual threads or processes over several nodes, a parallel program becomes a distributed one.

Erlang's message passing style fits the programming abstractions used in research on distributed systems. This enabled us to directly translate our abstract algorithms for transactions and the P2P layer into Erlang code. The whole transaction framework comprises only about 2,000 lines of code.

And Erlang's asynchronous message passing style is not the whole story either. Another powerful feature is its

standard library including OTP (Open Telecom Platform) which provides many useful abstractions for coping with failures. One example is the supervisors which are used to monitor and restart processes in case of node crashes.

Transaction API for Java and Erlang

The following Java code snippet illustrates the common bank account example where money is transferred from account A to B. The money transfer is bracketed within a transaction, thereby ensuring atomicity and isolation from other transactions.

```
// new Transaction object
Transaction transaction = new Transaction();

// start new transaction
transaction.start();

// read account A
int account_A = new Integer(transaction.read("account_A")).intValue();
// read account B
int account_B = new Integer(transaction.read("account_B")).intValue();

// remove 100$ from account A
transaction.write("account_A", new Integer(account_A - 100).toString());
// add 100$ to account B
transaction.write("account_B", new Integer(account_B + 100).toString());

// commit
transaction.commit();
```

The corresponding Erlang code is slightly more complex, because of the explicit handling of transaction states. The Erlang example below performs the same money transfer as the Java code.

```
TFun =
  fun(TransLog) ->
    % read balance
    {MyBalance, TransLog1}    = read2(TransLog,  MyAccount),
    {OtherBalance, TransLog2} = read2(TransLog1, OtherAccount),
    % update balance
    TransLog3 = write2(TransLog2, MyAccount,    MyBalance - 100),
    TransLog4 = write2(TransLog3, OtherAccount, OtherBalance + 100),
    {ok, TransLog4}
  end.

SuccessFun = fun(X) ->
                 {success, X}
             end.
FailureFun = fun(Reason) ->
                 {failure, Reason}
             end.

% execute transaction
do_transaction(TFun, SuccessFun, FailureFun).
```

The Erlang transaction API is more powerful and it exposes all functionality, including the asynchronous execution. Transactions are expressed as anonymous functions, i.e. pointers to functions like TFun. TFun is a function, which, when executed, records all reads and writes in a transaction log.

The function do_transaction first calls TFun to gather the transaction log (read phase) and then tries to atomically commit the recorded changes to the system (commit phase). If a concurrent write operation on one of the involved items is detected, the transaction is aborted and the user defined FailureFun is executed. Otherwise the given SuccessFun is called.

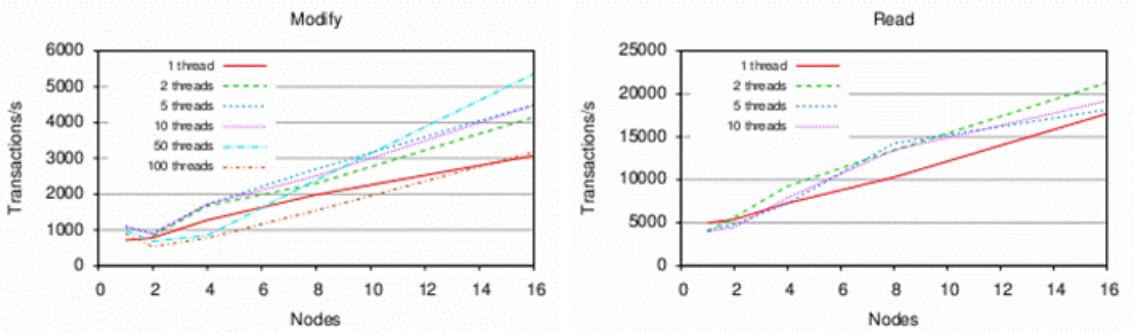
The Java API, in contrast, exposes only a subset of the functionality but is more convenient to use.

Demo Application

As a demonstrator application we implemented a subset of the Wikipedia functionality with Scalaris as the database. The user-facing webservers are standard Java Servlet containers which implement the application logic and render the WikiText to HTML.

The database backend implements a thin layer for mapping the Wikipedia SQL tables to our data model. The Scalaris data model is essentially equivalent to a Map<String,String> with support for range queries. For Wikipedia we had to split its relational scheme into key-value pairs.

Scalability



We tested the performance of Scalaris on an Intel cluster. Each node has two Quad-Core E5420s running at 2.5 GHz and 16GM of main memory. On each physical cluster node, we ran one Erlang virtual machine and 16 Scalaris nodes. The graphs show the performance of modify operations (top graph) and read operations (bottom graph) with a replication degree of four. The read operation reads a majority of the replicas of one key-value pair while the modify operation performs a read-modify-write cycle on a key-value pair in a transaction.

The graphs show the number of threads executing the benchmark per node and the number of cluster nodes used. Note, that for the 100-thread-case, there are actually up to 16*100 threads issuing modify transactions concurrently. The read as well as the modify scale almost linearly with the number of nodes. To achieve the best performance, more concurrent data accesses are needed in the modify operation compared to the read operation.

Additional Information

The Scalaris code is open source. It is available at http://code.google.com/p/scalaris/ . Additional information (papers, videos) can be found at http://www.onscale.de .

PRINTER FRIENDLY VERSION

TheServerSide.COM
Your Enterprise Java Community