The ETH Zurich Systems Group and Enterprise Computing Center

Gustavo Alonso Donald Kossmann Timothy Roscoe Nesime Tatbul
Andrew Baumann Carsten Binnig Peter Fischer Oriana Riva Jens Teubner
Systems Group
Department of Computer Science, ETH Zurich
Zurich 8092, Switzerland
http://www.systems.inf.ethz.ch/

1. INTRODUCTION

Computer science is facing a fundamental paradigm shift. *Multicore architectures*, *application virtualization*, and *cloud computing* each present on their own radical departures in the way software is built, deployed, and operated. Taken together, these trends frame a scenario that makes sense from many points of view (usability, scalability, flexibility, development cost) but is also radically different from what we know today. It is fair to say that a coherent answer to the challenges raised by the combination of these trends has yet to emerge from either academia or industry.

From an academic perspective, these challenges are particularly difficult because they imply a considerable departure from established procedures. To start with, multicore computers, the virtualization of computing platforms, and the replacement of the *one-computer-one-local-copy-of-a-program* approach by cloud computing each demand an interdisciplinary approach. In practice, it is likely that traditional disciplines such as operating systems, distributed systems, software engineering, or data management will require major revision as the boundaries between them become blurred.

A further challenge for both academic and (to some extent) industrial research is the impossibility of exploring the important design, architectural, and algorithmic challenges ahead using a small number of computers. Industrially meaningful systems today are large distributed platforms (hundreds, thousands, ten of thousands nodes) with complex multi-layered architectures, often geographically distributed over the globe. Academia often lacks both access to such systems and basic information on the operations, constraints and requirements involved.

The *Systems Group*, together with the *Enterprise Computing Center* (ECC) are two recent initiatives at the ETH Zurich Department of Computer Science to respond to these challenges. The goal of the Systems Group is to redefine, restructure, and reorganize systems research to avoid the pitfalls of looking at complex problems from a single, isolated perspective. The goal of the ECC is to establish new relationships between academia and industry that are longer term, more productive for all sides, and give academic research direct access to real systems and empirical data about the functioning of these systems.

In this short paper, we present both these initiatives and some of the associated research projects. We hope our ideas will inspire others; we are convinced that such research structures are essential for academic research to remain competitive and relevant in todays computing environment.

2. VISION AND APPROACH

Our vision of how mainstream computing is evolving is based on the three trends mentioned above: multicore, virtualization, and cloud computing. To these, we add *pervasive computing*, and note that the most common information access terminals in the near future will be portable devices rather than traditional computers.

We summarize the vision as follows: software will run on *manycore* (>16 cores) computers with *heterogeneous hardware* (not all cores and processing units will have the same capabilities). Applications will be deployed on clusters of thousands if not millions of machines, distributed worldwide. The hardware resources of these clusters will be *virtualized* into logical, dynamically configured computing platforms. Through virtualization and the notion of *software as a service*, applications on such platforms will operate in a *computing cloud:* physically separated from the application client or human user). The principle access device for the cloud will be future *portable computing and communication devices*, of which today's mobile phones are a precursor.

Our research agenda revolves around the many related challenges in moving from where we are and what we know today towards this emerging vision of computing:

1. Managing resources in a heterogeneous multicore computer that itself increasingly resembles a distributed system, and architecting applications (specially data management applications) to efficiently exploit hetergeneous multicore machines.

2. Architecting applications to efficiently exploit thousands of hetergeneous multicore machines, and building software platforms (database systems, data stream processors, application servers) on this hardware infrastructure.

3. Evolving existing software systems into multicore applications, and determining which parts of standard applications can run on specialized, dedicated hardware and which are the appropriate abstractions for programming such hardware.

4. Appropriate methodologies for developing, deploying, and maintaining modern applications in the cloud, and building application modules that can be automatically configured and dynamically managed in a virtual environment.

5. Determining which new classes of applications are possible in such environments.

6. Seamlessly integrating portable computing and communication devices and applications running in the computing cloud.

We approach these challenges by building *real, complete systems* as well as designing new algorithms, data structures, and protocols. In doing so, we try to inhabit this new world of large-scale computing as much as possible. However, this in turn cannot be done piecemeal at the level of individual research projects; it requires a substantial, long-term commitment of both people and resources. The Systems Group and the Enterprise Computing Center are a means to sustain such a research agenda.

3. THE SYSTEMS GROUP

Practical pursuit of our research vision requires a *critical mass* of people exploring and constructing systems, and the necessary *complementary expertise* to tackle all key aspects of the problem.

We formed the Systems Group by merging the research groups of four professors in different areas: Gustavo Alonso (distributed systems, middleware), Donald Kossmann (data management, databases), Timothy Roscoe (operating systems, networking), and Nesime Tatbul (data streams, distributed data management). The Group today has five senior researchers and 28 Ph.D. students.

Unlike many “laboratories” or “institutes”, the group truly functions as a single unit. For example, all students are assigned at least two professors in the group as advisors, based on the area they are working in. By having several real advisors (who actually act as such), students are forced from day one to explain their ideas to people from different areas and to look at their own work from varied perspectives.

Students collaborate in teams on the development of particular systems. Interaction across the group takes place in individual meetings (discussing the work of the individual student), project meetings (discussing the work on a project), strategic meetings (discussing the interactions across systems and projects), and at presentations and discussions where the whole group is involved. These meetings promote the constant exchange of ideas from networks, operating systems, distributed systems, and databases systems. They are also the source of new ideas and solutions that give us the necessary leverage to tackle many of the research questions listed above.

While such an interdisciplinary approach is not new in many universities, we have made it a mandatory part of the doctoral studies. There is naturally an intrinsic cost to this, since it takes longer to develop a common understanding of the problem at hand (simply because more angles are being considered at the same time). However, the effort is worthwhile: in two years of operation, we have many examples of successful, creative, and innovative projects generated from this process.

4. ENTERPRISE COMPUTING CENTER

Any research institution faces the problem of obtaining adequate funding while maintaining the freedom to pursue a coherent agenda. In our case, we are privileged to work at ETH Zurich, a university explicitly created for top-tier education *and* research. The generous funding available is a reflection of Switzerland’s commitment to research, and the understanding that research must have substantial independence from industry and other external agencies. The base funding provided for each professor allowed us to create the Systems Group while neither compartmentalizing our research agenda nor tailoring it to the vagaries of any funding agency. This is particularly important since many of the challenges we are pursuing are not yet well-formulated in the way required by some funding sources – in fact, part of our work is to formulate these problems more precisely.

However, even in the generous research environment that exists at ETH Zurich, systems work in academia increasingly requires access to large computing and software infrastructures beyond the means of a single institution (a situation that has lead to the recent creation of shared research platforms such as PlanetLab [9]). Our research agenda requires an understanding of, and access to, computer systems used in industry that simply cannot be replicated at a university. The Enterprise Computing Center was created to address this and provide a vehicle for validating ideas and prototypes against real systems. Besides helping to fund our research, it creates an open and ongoing dialog between us and industry that speeds up tech transfer and keeps the research focused and honest. The ECC brings the insights of industry to academic research without actually tying the work to particular products or corporate strategies.

The ECC model is based on long-term collaboration between industry and academia, sustained through close interaction. Rather than collaboration on a per-project basis, the ECC establishes a framework in which to discuss research opportunities and to define concrete projects that are of mutual interests to all parties involved. This requires industrial partners to commit time and people to ECC, not just funds. It typically happens when both the Systems Group and the industrial partner have established that there are areas of common interest, and the long term research carried out by the Systems Group can be of value to the company. Concrete projects can then be defined and carried out under several possible models: students working mostly at ETH Zurich, students working mostly at the industrial partner, or a hybrid approach where frequent visits between partners facilitate the tech transfer. In all cases, the students are full Ph.D. students at ETH Zurich, advised by ETH professors and subject to the same requirements and standards than any other Ph.D. student. The ECC and the research projects conducted through it are fully integrated into the rest of the Systems Group. With regard to IP, the IP is owned by both ETH and the industry partner if not specified otherwise; if the company wants to own all the IP, the company must pay overheads to ETH accordingly. In all cases, it is guaranteed by the ETH contracts that the students may publish all research results.

The ECC also provides a common ground for different companies to meet and develop a common understanding of problems faced across industries. We organize an annual workshop with all partners to discuss the project progress, identify avenues for further research, and exchange ideas and results. The first such workshop took place in November, 2008 at Monte Verita, in Ascona, Switzerland.

The current ECC partners are Credit Suisse, Amadeus, and SAP. All are interested in several of the research challenges above and were already individually exploring different versions of our vision. These partners have complementary interests, do not compete in the market, and are committed to an open research collaboration, a principle we intend to maintain as more partners are added.

5. RESEARCH PROJECTS

To concretize the discussion, we describe a collection of inter-related projects being pursued in the Group. Some are under the auspices of the ECC, while some involve other industry collaborations. We make a somewhat arbitrary classification into multicore machine-related projects, building blocks for future applications, and finally cloud computing.

5.1 Multicore and New Platforms

This first group of projects is about exploiting and managing resources at the level of single machines, as a basis for the higher layers of both applications and the distributed infrastructure they will run on.

5.1.1 Operating systems: Barrelfish

Barrelfish is a new operating system developed in the Systems Group in collaboration with Microsoft Research in Cambridge, UK. Barrelfish is open source, written largely from scratch, and targets emerging hetergeneous multicore systems and the application runtimes being developed for them.

Barrelfish is a reaction to challenges both from application software and emerging hardware. In software, increasingly sophisticated languages and runtimes are appearing to allow programmers to effectively express parallelism in their programs, ranging from parallel combinators in functional languages to re-architected database query engines, but it is not clear that current I/O APIs for operating systems like Windows and POSIX can efficiently support such systems without the OS becoming the bottleneck.

In hardare, the increasing number of CPU cores is accompanied by an increasing diversity in computer system architectures: machines themselves are becoming more diverse, heterogeneous cores within a machine are becoming the norm, and memory systems are increasingly non-uniform. Current OS architectures are ill-equipped to exploit such a complex and varying feature set without an explosion in code complexity.

Barrelfish applies two key insights to these challenges. Firstly, the machine is treated for the most part as a distributed system in its own right: message passing and agreement protocols are used wherever possible between cores rather shared memory to reduce expensive cross-system locks and contention for both the memory system and interconnect.

Secondly, we apply techniques from data management and knowledge representation to allow the OS and applications to reason effectively about the machine and optimize their policies accordingly [8]. Barrelfish includes a constraint logic programming (CLP) solver as a basic system service, in place of the name services or configuration databases found in conventional systems. We hope the powerful combination of logical inference and constrained optimization will allow effective exploitation of a wide range of current and future hardware.

5.1.2 Hardware acceleration: NICs, FPGAs

To explore the potential of specialized hardware in tandem with Barrelfish, we are studying how to use programmable Network Interface Cards (NICs) and Field Programmable Gate Arrays (FPGAs) to speed up specialized operations, with an initial focus on stream processing.

In the context of programmable NICs, we are studying the acceleration of message processing as part of algorithmic trading in conventional financial applications. The challenge here is to minimize the latency between the arrival of a message and the reaction to it.

We are also working on implementing conventional data stream operators in FPGAs, as a way to determine the sweet spot for this technology. FPGAs are particularly interesting because they allow the development of hardware tailored at runtime, and can be embedded into a CPU socket in multi-socket PC motherboards.

A longer-term goal of both these efforts is to explore how Barrelfish can be used to manage and allocate such heterogeneous resources, and how applications can be built to efficiently exploit such heterogeneous platforms.

This work is partially in collaboration with Credit Suisse, and also supported in part by an IBM Faculty Award to Nesime Tatbul.

5.1.3 Software architecture: Universal OSGi

The view of the future we subscribe to implies the need for frameworks for developing, deploying, and managing application over manycore and cluster-based systems. The Universal OSGi project (funded in part by the Microsoft Research Innovation Cluster in Software for Embedded Systems) aims to simplify software development software over both multicore and virtualized clusters. We build on the well-understood concept of software module, in particular as embodied in the Java-based Open Services Gateway Initiative (OSGi) standard. The key idea is to abstract complex tasks such as deployment of distributed behind traditional module composition and life-cycle management. Programmers develop modules and the platform uses these modules for distributed deployment in a cluster or for parallelization on a multicore machine.

We have already validated the basic idea in practice in two implementations of the OSGi specification. The first, Concierge (http://concierge.sourceforge.net/) [10], is a high-performance, low footprint implementation for small devices. The second is R-OSGi [11], which extends the OSGi model to support transparent distribution of an application across different nodes on a network, by mapping remove invocations and parial failure to module calls and module unload events respectively. R-OSGi hides the difficulties of developing distributed software by basing the distribution on the modular architecture of the application.

The next step is Universal OSGi: applying these ideas outside the Java world. We have taken the first steps in this direction by showing how the model of R-OSGi can be used as fabric for the so-called Internet of Things [12], including extensions to treat code developed in languages other than Java as OSGi bundles, turning them into components with the same characteristics as any other component within an OSGi framework. We are also developing a C analogue of

OSGi, one use of which is as the object binding model in Barrelfish.

5.1.4 *Multicore query processing: CresCanDo*

CresCanDo is a collaboration between the Systems Group and Amadeus as part of the ECC to provide predictable performance for unpredictable query and update workloads. The key idea is to rely on collaborative scans in main-memory as the only access path for query processing. Scalability within a node comes from running each scan on a separate core of a multicore machine. Scalability as a whole comes from using a large enough cluster of such machines to keep all data in main memory. The crucial advantage of this approach is that the query and update response-times become predictable, independent of any indexing. Horizontally partitioning the database and executing all scans in main memory results in manageable response times. To handle high update rates, a relaxed consistency model is employed which does not support serializability but gives strong guarantees on freshness of data scanned for a given query.

The contributions of the project include a novel algorithm to schedule and process large main-memory scans for queries and updates on multicore machines [14]. We have also developed a new logging and local recovery mechanism that persists all data to disk with little overhead and recovers a machine after a crash with reasonable effort. While CresCanDo has a clear direction of its own and is already being tested at Amadeus, it is also a good use case for other projects. CresCanDo will be ported to Barrelfish as the most natural OS for such an application, and we are also exploring using Remote Direct Memory Access (RDMA) for fast recovery strategies for CresCanDo.

5.2 Building Blocks

At a higher level, a number of projects in the systems group are exploring building blocks for large-scale services. Each project is focussed on a concrete, independent result, but these results form important components of the wider vision.

5.2.1 *Storage management for streams: SMS*

Flexible and scalable storage management is a key issue in the performance of data-intensive streaming applications. The SMS project (funded by the Swiss National Science Foundation) proposes a stream processing architecture that decouples query processing from storage management to flexibly allow fine-tuning of storage mechanisms according to application needs. We first define a parametric interface between the query processor and the storage manager, general enough to capture the architectural, functional, and performance-related properties of the most common set of streaming applications. In particular, data access patterns for reads and updates have the most direct impact on performance. By analyzing the possible forms of these access patterns we devise a set of corresponding data structures, access paths, and indices to minimize overall memory consumption and application query response time. The SMS interface also facilitates multi-query optimization based on the shared access patterns of multiple queries. A recent study has shown the performance benefits of the SMS approach [1].

In addition to supporting performance improvements, SMS also provides a clean and flexible system architecture that we have found useful as a building block in other projects. SMS is the underlying storage manager in XTream, and both the UpStream and DejaVu projects build on the design for application-specific performance tuning. Finally, we believe that SMS-style loose-coupling is also appropriate to federated stream processing and data management in the cloud.

5.2.2 *Interfacing to the cloud: AlfredO*

Like many others, we believe mobile devices will be the main access point to the cloud for many end users. Seamlessly integrating code on such devices with applications in the cloud is therefore a key challenge. Our first steps in this direction include AlfredO [13]: a lightweight middleware architecture that enables users to easily interact with other systems while providing ease of maintenance and security for their personal devices.

AlfredO is based on two insights. The first is that interactions with devices like appliances, touch-screens, vending machines, etc., tend to be short-term and ad-hoc, and so the traditional approach of pre-installing drivers or interface code for each target device is impractical. Instead, we employ a distribution model based on the idea of software as a service: each target device presents its capabilities as modular service items that can be accessed on-the-fly by a personal device such as a phone.

The second insight derives from the evolution of client-server computing from mainframes and terminals, through two-tier (client-server) systems, to three-tier architectures such as Web applications: partitioning server functionality leads to better performance, scalability, flexibility, and adaptability. We model each service in the cloud as a decomposable multi-tier architecture consisting of presentation, logic, and data tiers.

These tiers can be selectively distributed to the user's mobile device through AlfredO depending on the optimal configuration for the task at hand. AlfredO makes extensive use of R-OSGi (part of the Universal R-OSGi project). Initial testing has been possible thanks to a generous equipment grant from Nokia.

Our current work is to use AlfredO in conjunction with the Rhizoma runtime, employing a mobile phone as an interface to compute-intensive, real-time "recognition, mining and synthesis" workloads such as 3D scene generation that cannot be performed purely on the phone.

5.2.3 *Zorba and MXQuery*

In a recent Communications of the ACM article [6], the plethora of programming languages and technologies needed to build a large-scale application was identified as a major limiting factor for cloud computing. Independent of cloud computing, simplifying the software stack is an important part of improving the development, evolvability, and deployment of applications. Today, SQL is the dominant programming language at the database layer, Java (or C# or other object-oriented languages) in the application layer, and JavaScript is the language of choice in the presentation layer (the browser).

We are investigating the use of a single programming language and server software architecture for all application tiers. This would allow great flexibility in deployment, allowing application code to move between the client and the server, or be pushed down to the database. Furthermore, a "whole program" view would make such applications more amenable to automatic optimization. Finally, data marshalling between layers becomes more uniform and in some cases can be eliminated entirely, and the replication of functionality (such as integrity checking or logging) across layers is avoided.

We have started by developing pluggable processors for the XQuery language: Zorba and MXQuery. XQuery seems to be a good match for database queries, application logic, and user interfaces in the Web browser, and has recently acquired extensions for REST, Web Services, and window-based stream processing. While XQuery's status as a W3C standard makes it a natural choice, other languages such as Microsoft's LINQ are also good candidates; our principle interests are language-independent and rather concern the development of new architectures for stream processing [2] (also used in the XTream project), browser programming [5], and application servers for the cloud [7].

5.3 Cloud Computing and Virtualization

Our final group of projects builds on the technologies we have just described, to facilitate the design and implementation of complete applications and software services deployed on cloud infrastructures.

5.3.1 *Data management in the cloud: Cloudy*

Despite the potential cost advantages, cloud-based implementations of the functionality found in traditional databases face significant new challenges, and it appears that traditional database architectures are poorly equipped to operate in a cloud environment.

For example, a modern database system generally assumes that it has control over all hardware resources (so as to optimize queries) and all requests to data (so as to guarantee consistency). Unfortunately, this assumption limits scalability and flexibility, and does not correspond to the cloud model where hardware resources are allocated dynamically to applications based on current requirements. Furthermore, cloud computing mandates a loose coupling between functionality (such as data management) and machines. To address these challenges, we are developing a system called Cloudy [3, 4], a novel architecture for data management in the cloud. Cloudy is a vehicle for exploring design issues such as relaxed consistency models and the cost efficiency of running transactions in the cloud.

We are also rethinking the model for distributed and potentially long-running transactions across autonomous services (such as those found in the cloud). One key idea is to employ a reservation pattern in which updates are reserved before they are actually committed – in some sense, a generalization of 2-phase commit in which the ability to commit is *reserved* before the actual commit itself. We are exploring this pattern in collaboration with Oracle and Credit Suisse so as to understand its domain of applicability for large-scale applications and complex infrastructures.

5.3.2 *Self-deploying applications: Rhizoma*

Data management is only one challenge posed by deploying long-running services on cloud infrastructures. Selecting cloud providers is becoming more complex as more players enter the market, pricing structures change regularly through competition and innovation, individual providers experience transient failures and major outages, and application deployment must be adjusted (within constraints) to handle changes in offered load.

Rhizoma [15] explores a novel approach to such challenges. Instead of the additional complexity and overhead of using a management console or service separate from the application, we bundle a management runtime with the application which can acquire new resources and deploy further application instances as needed, with no separate management infrastructure required.

A distributed Rhizoma application can span multiple cloud providers, and is almost entirely autonomous: individual application nodes elect a leader that monitors resource availability and usage, decides on future resource requirements, acquires and releases virtual machines as required, and deploys new instances of the application where needed.

Developers or service operators specify the policy for deployment of a Rhizoma application as a high-level constrained optimization problem (such as maximizing available CPU while minimizing overlay network diameter and monetary cost), which is used by the leader to make deployment decisions on a continuous basis.

We are currently considering using Rhizoma in a variety of other projects: in combination with Universal OSGi, and as an extension to XTream and AlfredO.

5.3.3 *Federated stream processing: MaxStream*

Despite the availability of several data stream processing engines (SPEs) today, it remains hard to develop and maintain streaming applications. One difficulty is the lack of agreed standards, and the wide (and changing) variety application requirements. Consequently, existing SPEs vary widely in data and query models, APIs, functionality, and optimization capabilities. Furthermore, data management for stored and streaming data are still mostly separate concerns, although applications increasingly require integrated access to both. In the MaxStream project, our goal is to design and build a federated stream processing architecture that seamlessly integrates multiple autonomous and heterogeneous SPEs with traditional databases, and hence facilitates the incorporation of new functionality and requirements.

MaxStream is a federation layer between client applications and a collection of SPEs and databases. A key idea is to present at the application layer a common SQL-based query language and programming interface. The federation layer performs global optimizations and necessary translations to the native interfaces of the underlying systems. The second idea is to implement the federation layer itself using a relational database infrastructure. By doing so, we can build on existing support for SQL, persistence, transactions, and most importantly traditional federation functionality. Finally, MaxStream leverages the strengths of the underlying engines while the federation layer can compensate for any missing functionality by itself adding a number of novel streaming features on top of the relational engine infrastructure. MaxStream is a collaboration with SAP Labs in the context of the ECC, and also builds on the SMS storage manager project.

5.3.4 *Global stream overlays: Xtream*

The XTream project is looking at stream processing as the basis for a global scale, collaborative data processing and dissemination platform where independent processing units are linked by channels to form intertwined data stream processing meshes. In XTream we seek to generalize the data stream processing model beyond current applications (stock tickers, sensor data, etc.) to a more general class of pervasive streaming applications encompassing a wider range of heterogeneous information sources and forms of data exchange (e-mail, messaging, SMSs, notifications, alarms, pictures, events, etc.).

XTream has been designed as a dynamic and highly distributed mesh of data processing stages connected through strongly typed channels, which connect heterogeneous data sources and sinks through standard interfaces and support in-network data processing and storage. Stages export standard interfaces, while the channels provide an underlying storage and messaging fabric.

The mesh overlay is extensible and configurable at runtime: stages and channels can be dynamically added and removed, with the system ensuring continuous operation and consistent results during this process. Through XTream we are exploring fundamental design questions for highly distributed systems and how to bring stringer software engineering design concepts into system architectures.

XTream is funded in part by the Swiss National Science Foundation. It builds upon R-OSGi (part of the Universal OSGi project) and the SMS project, and also serves within the group as a general use-case for large scale pervasive computing using clouds.

6. CONCLUSION

The experience we have accumulated in the last two years with The Systems Group and the Enterprise Computing Center has been overwhelmingly positive. The advantages more than compensate for the intrinsic coordination and communication cost of a larger working unit, a view shared by all involved from faculty to PhD students. The Enterprise Computing Center has also become a crucial part of our research, with projects that not only are at the forefront of technology but also bring first hand feedback from industry and have an open door for technology transfer. More information on the group, ECC, or any of our research or teaching activities can be found in our web pages (`http://www.systems.inf.ethz.ch/`). Those interested in pursuing Master studies at ETHZ, doing a PhD within the Systems Group, a Post-Doc position, or spending time as a faculty visitor, should contact the faculty by e-mail.

7. ACKNOWLEDGMENTS

In addition to the support we have received from ETH Zurich, we would also like to thank all our various sponsors and collaborators, including Amadeus, Credit Suisse, the European Commission, the FLWOR Foundation, the Hasler Foundation, IBM, Intel, Microsoft, Nokia, Oracle, SAP, and the Swiss National Science Foundation (SNF). Finally, we would also like to thank all the members of the Systems Group for making it such a fun and successful place to work.

8. REFERENCES

[1] I. Botan, G. Alonso, P. M. Fischer, D. Kossmann, and N.Tatbul. Flexible and Scalable Storage Management for Data-intensive Stream Processing. In *International Conference on Extending Database Technology (EDBT'09)*, Saint Petersburg, Russia, March 2009.

[2] I. Botan, P. Fischer, D. Florescu, D. Kossmann, T. Kraska, and R. Tamosevicius. Extending XQuery with Window Functions. In *Proceedings of VLDB 2007*, Vienna, Austria, September 2007.

[3] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska. Building a database on S3. In *Proceedings of the ACM SIGMOD Conference*, Vancouver, Canada, June 2008.

[4] D. Florescu and D. Kossmann. Rethinking the cost and performance of database systems. `http://www.dbis.ethz.ch/research/publications/index`, December 2008.

[5] G. Fourny, D. Kossmann, T. Kraska, M. Pilman, and D. Florescu. XQuery in the browser - Demo paper. In *Proceedings of the ACM SIGMOD Conference*, Vancouver, Canada, June 2008.

[6] B. Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

[7] D. Kossmann. Building Web Applications without a Database System - Invited Talk. In *Proceedings of the EDBT 2008 Conference*, Nates, France, March 2008.

[8] S. Peter, A. Schüpbach, A. Singhania, A. Baumann, T. Roscoe, P. Barham, and R. Isaacs. Multikernel: An architecture for scalable multi-core operating systems (Work-in-Progress report). In *Proceedings of OSDI 2009*, San Diego, CA, USA, December 2008.

[9] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, USA, October 2002.

[10] J. Rellermeyer and G. Alonso. Concierge: A Service Platform for Resource-Constrained Devices. In *Proceedings of the ACM EuroSys 2007 Conference*, Lisbon, Portugal, March 2007.

[11] J. Rellermeyer, G. Alonso, and T. Roscoe. R-OSGi: Distributed Applications through Software Modularization. In *Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)*, Newport Beach, CA, USA, November 2007.

[12] J. Rellermeyer, M. Duller, K. Gilmer, D. Maragkos, D. Papageorgiou, and G. Alonso. The Software Fabric for the Internet of Things. In *Proceedings of the First International Conference on the Internet of Things*, Zurich, Switzerland, March 2008.

[13] J. Rellermeyer, O. Riva, and G. Alonso. AlfredO: An Architecture for Flexible Interaction with Electronic Devices. In *Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference (Middleware 2008)*, Leuven, Belgium, December 2008.

[14] P. Unterbrunner, G. Giannikis, G. Alonso, D. Fauser, and D. Kossmann. Clockscan: Predictable performance for unpredictable workloads. Technical Report, ETH Zurich, in preparation, 2009.

[15] Q. Yin, J. Cappos, A. Baumann, and T. Roscoe. Dependable Self-Hosting Distributed Systems Using Constraints. In *Proceedings of the 4th Usenix Workshop on Hot Topics in System Dependability (HotDep)*, San Diego, CA, USA, December 2008.