IN PARTNERSHIP WITH:
CNRS

Ecole des Mines de Nantes

IMT Atlantique Bretagne-Pays de la Loire

Université Nantes

Activity Report 2016

Project-Team ASCOLA

Aspect and composition languages

IN COLLABORATION WITH: Laboratoire des Sciences du numerique de Nantes

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Distributed programming and Software engineering

Table of contents

1. Members **2**
2. Overall Objectives **3**
3. Research Program **3**
3.1. Overview 3
3.2. Software Composition 4
3.3. Programming languages for advanced modularization 4
3.4. Distribution and Concurrency 6
3.5. Security 6
3.6. Green IT 7
3.7. Capacity Planning for Large Scale Distributed System 7
4. Application Domains **8**
4.1. Enterprise Information Systems and Services 8
4.2. Capacity Planning in Cloud, Fog and Edge Computing 8
4.3. Pervasive Systems 9
5. Highlights of the Year **9**
6. New Software and Platforms **9**
6.1. CSLA 9
6.2. CSQL 10
6.3. EScala 10
6.4. JEScala 11
6.5. SimGrid 11
6.6. VMPlaces 11
6.7. btrCloud 12
7. New Results **12**
7.1. Software composition and programming languages 12
7.1.1. Formal Methods, logics and type theory 13
7.1.1.1. Verified Dependent Interoperability. 13
7.1.1.2. Forcing in Type Theory. 13
7.1.2. Programming languages 13
7.1.2.1. Constraint programming 13
7.1.2.2. Program correctness 14
7.1.2.3. Effect Capabilities 14
7.1.2.4. Extensible JavaScript Modules 14
7.1.3. Software Security and Privacy 14
7.1.3.1. Runtime verification of advanced logical security properties. 14
7.1.3.2. Specification of advanced security and privacy properties. 15
7.1.3.3. Composition of privacy-enhancing and security mechanisms. 15
7.2. Distributed programming and the Cloud 15
7.2.1. Cloud applications and infrastructures 15
7.2.1.1. Service-level agreement for the Cloud. 16
7.2.1.2. Cloud Capacity Planning and Elasticity. 16
7.2.1.3. Infrastructure. 16
7.2.2. Renewable energy 17
8. Bilateral Contracts and Grants with Industry **17**
9. Partnerships and Cooperations **18**
9.1. Regional Initiatives 18
9.1.1. RFI Atlanstic 2020 18
9.1.2. Pays de la Loire 18
9.2. National Initiatives 18

9.2.1. CominLabs laboratory of excellence 18
9.2.1.1. EPOC 18
9.2.1.2. PrivGen 19
9.2.1.3. SecCloud 19
9.2.2. ANR 20
9.2.3. FSN 20
9.2.3.1. OpenCloudware (FSN) 20
9.2.3.2. Hosanna (FSN) 20
9.2.4. CPER 20
9.2.5. Inria Project Labs 21
9.2.6. InriaHub 21
9.3. European Initiatives 22
9.3.1.1. CoqHoTT 22
9.3.1.2. BigStorage 22
10. Dissemination **23**
10.1. Promoting Scientific Activities 23
10.1.1. Scientific Events Organisation 23
10.1.2. Scientific Events Selection 23
10.1.3. Journal 23
10.1.3.1. Member of the Editorial Boards 23
10.1.3.2. Reviewer - Reviewing Activities 23
10.1.4. Invited Talks 24
10.1.5. Leadership within the Scientific Community 24
10.1.6. Research Administration 24
10.2. Teaching - Supervision - Juries 24
10.2.1. Teaching 24
10.2.2. Juries 25
10.3. Popularization 25
11. Bibliography **25**

Project-Team ASCOLA

Creation of the Project-Team: 2009 January 01

Keywords:

Computer Science and Digital Science:

1.1.6. - Cloud
1.1.8. - Security of architectures
1.1.13. - Virtualization
1.3. - Distributed Systems
1.6. - Green Computing
2.1. - Programming Languages
2.1.1. - Semantics of programming languages
2.1.2. - Object-oriented programming
2.1.3. - Functional programming
2.1.4. - Aspect-oriented programming
2.1.6. - Concurrent programming
2.1.7. - Distributed programming
2.1.10. - Domain-specific languages
2.1.11. - Proof languages
2.2.1. - Static analysis
2.4.2. - Model-checking
2.4.3. - Proofs
2.5. - Software engineering
2.6.2. - Middleware
2.6.3. - Virtual machines
3.1.3. - Distributed data
3.1.5. - Control access, privacy
3.1.8. - Big data (production, storage, transfer)
4.5. - Formal methods for security
4.6. - Authentication
4.7. - Access control
4.8. - Privacy-enhancing technologies
7.1. - Parallel and distributed algorithms
7.4. - Logic in Computer Science

Other Research Topics and Application Domains:

3.1. - Sustainable development
4.5. - Energy consumption
4.5.1. - Green computing
5.1. - Factory of the future
6.1. - Software industry
6.1.1. - Software engineering
6.1.2. - Software evolution, maintenance
6.5. - Information systems

1. Members

Research Scientists

Adrien Lebre [Inria, Researcher, on leave from MN]
Guillaume Munch [Inria, Researcher, from Oct 2016]
Nicolas Tabareau [Inria, Researcher]

Faculty Members

Mario Südholt [Team leader, MN, Professor, HDR]
Frederico Alvares de Oliveira Junior [MN, Associate Professor, temporary position]
Julien Cohen [Univ. Nantes, Associate Professor]
Pierre Cointe [MN, Professor, HDR]
Hélène Coullon [MN, Associate Professor, Inria chair, from Oct 2016]
Rémi Douence [MN, Associate Professor, HDR]
Hervé Grall [MN, Associate Professor]
Thomas Ledoux [MN, Associate Professor]
Jean-Marc Menaud [MN, Professor, HDR]
Jacques Noyé [MN, Associate Professor]
Jean-Claude Royer [MN, Professor, HDR]

Engineers

Ronan-Alexandre Cherrueau [Inria, from Oct 2016]
Rémy Pottier [MN]

PhD Students

Mohamed Abderrahim [Orange Labs, granted by CIFRE]
Mohammad-Mahdi Bazm [Orange Labs, from Jul 2016]
Walid Benghabrit [ARMINES, until Sep 2016]
Paul Blouët [MN]
Simon Boulier [MN, (ASN from ENS Rennes)]
Bastien Confais [CNRS, from Oct 2016]
Ismael Cuadrado Cordero [MN]
Frédéric Dumont [EasyVirt, until Jun 2016, granted by CIFRE]
Simon Dupont [Sigma, until Apr 2016, granted by CIFRE]
Alexandre Garnier [MN, until Sep 2016]
Gaetan Gilbert [MN (ASN from ENS Lyon), from Sept 2016]
Md Sabbir Hasan [Inria, co-supervision with Prof. Pazat, Myriads team, Inria]
Yacine Hebbal [Orange Labs, granted by CIFRE]
Ambroise Lafont [MN (X Grant), from Oct 2016]
Gabriel Lewertowski [Inria, until Oct 2016]
Florent Marchand de Kerchove [MN, until Apr 2016]
Thuy Linh Nguyen [Inria]
Jonathan Pastor [MN, until March 2016]
Kevin Quirin [MN]

Post-Doctoral Fellows

Benedikt Ahrens [Inria, from May 2016]
Zakarea Al Shara [MN, from Oct 2016]
Ali Kassem [Inria]
Guillaume Le Louet [MN, until Jul 2016]
Pierre-Marie Pedrot [Inria, until Sep 2016]
Dimitri Pertin [Inria, from Oct 2016]
Anthony Simonet [Inria]

Visiting Scientists

Paige North [Inria, from Sep 2016]
Egbert Rijke [Carnegie Mellon Uni., from Jun 2016 until Jul 2016]

Administrative Assistants

Anne Claire Binétruy [Inria, (part time 30%)]
Florence Rogues [MN, (part time 30%)]

Others

Marie Delavergne [Inria, Master student, from Apr 2016 until Aug 2016]
Theo Winterhalter [ENS Cachan, Master student, from Mar 2016 until Aug 2016]

2. Overall Objectives

2.1. Presentation

The research team addresses the general problem of evolving software by developing concepts, languages, implementations and tools for building software architectures based on components and aspects. Its long term goal is the development of new abstractions for the programming of software architectures, their representation in terms of expressive programming languages and their correct and efficient implementation.

We pursue the following objectives:

- New concepts and techniques for the compositional definition and implementation of complex software systems, notably involving crosscutting concerns that cannot be handled modularly using traditional software development approaches.
- New programming techniques and algorithms for resource management in mutualized environments. We provide language abstractions and implementation techniques for large-scale applications in cloud- and grid-based systems, both on the level of (service-based) applications and (virtualized) infrastructures. We develop solutions, in particular, for the optimization of the energy consumption in such environments (data centers ...)
- We develop new formal theories for and apply formal methods to the correctness of software systems. We aim at developing more powerful techniques for theorem proving and enable complex, often dynamic, software systems to be proven correct using program transformations and analysis techniques. We develop solutions, in particular, for the constructive enforcement of security properties on the level of software systems.

Finally, we apply and validate our results based on real-world applications from numerous domains, notably enterprise information systems, the Cloud, and pervasive systems.

3. Research Program

3.1. Overview

Since we mainly work on new concepts for the language-based definition and implementation of complex software systems, we first briefly introduce some basic notions and problems of software components (understood in a broad sense, that is, including modules, objects, architecture description languages and services), aspects, and domain-specific languages. We conclude by presenting the main issues related to distribution and concurrency, in particular related to capacity planning issues that are relevant to our work.

3.2. Software Composition

Modules and services. The idea that building *software components*, i.e., composable prefabricated and parameterized software parts, was key to create an effective software industry was realized very early [72]. At that time, the scope of a component was limited to a single procedure. In the seventies, the growing complexity of software made it necessary to consider a new level of structuring and programming and led to the notions of information hiding, *modules*, and module interconnection languages [79], [55]. Information hiding promotes a black-box model of program development whereby a module implementation, basically a collection of procedures, is strongly encapsulated behind an interface. This makes it possible to guarantee logical invariant *properties* of the data managed by the procedures and, more generally, makes *modular reasoning* possible.

In the context of today's Internet-based information society, components and modules have given rise to *software services* whose compositions are governed by explicit *orchestration or choreography* specifications that support notions of global properties of a service-oriented architecture. These horizontal compositions have, however, to be frequently adapted dynamically. Dynamic adaptations, in particular in the context of software evolution processes, often conflict with a black-box composition model either because of the need for invasive modifications, for instance, in order to optimize resource utilization or modifications to the vertical compositions implementing the high-level services.

Object-Oriented Programming. Classes and objects provide another kind of software component, which makes it necessary to distinguish between *component types* (classes) and *component instances* (objects). Indeed, unlike modules, objects can be created dynamically. Although it is also possible to talk about classes in terms of interfaces and implementations, the encapsulation provided by classes is not as strong as the one provided by modules. This is because, through the use of inheritance, object-oriented languages put the emphasis on *incremental programming* to the detriment of modular programming. This introduces a white-box model of software development and more flexibility is traded for safety as demonstrated by the *fragile base class* issue [75].

Architecture Description Languages. The advent of distributed applications made it necessary to consider more sophisticated connections between the various building blocks of a system. The *software architecture* [84] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [43]. It also describes the rationale behind such a given architecture, linking the properties required from the system to its implementation. *Architecture Description Languages* (ADLs) are languages that support architecture-based development [73]. A number of these languages make it possible to generate executable systems from architectural descriptions, provided implementations for the primitive components are available. However, guaranteeing that the implementation conforms to the architecture is an issue.

Protocols. Today, protocols constitute a frequently used means to precisely define, implement, and analyze contracts, notably concerning communication and security properties, between two or more hardware or software entities. They have been used to define interactions between communication layers, security properties of distributed communications, interactions between objects and components, and business processes.

Object interactions [77], component interactions [90], [81] and service orchestrations [56] are most frequently expressed in terms of *regular interaction protocols* that enable basic properties, such as compatibility, substitutability, and deadlocks between components to be defined in terms of basic operations and closure properties of finite-state automata. Furthermore, such properties may be analyzed automatically using, e.g., model checking techniques [53], [62].

However, the limited expressive power of regular languages has led to a number of approaches using more expressive *non-regular* interaction protocols that often provide distribution-specific abstractions, e.g., session types [66], or context-free or turing-complete expressiveness [82], [50]. While these protocol types allow conformance between components to be defined (e.g., using unbounded counters), property verification can only be performed manually or semi-automatically.

3.3. Programming languages for advanced modularization

The main driving force for the structuring means, such as components and modules, is the quest for clean *separation of concerns* [57] on the architectural and programming levels. It has, however, early been noted that concern separation in the presence of crosscutting functionalities requires specific language and implementation level support. Techniques of so-called *computational reflection*, for instance, Smith's 3-Lisp or Kiczales's CLOS meta-object protocol [85], [69] as well as metaprogramming techniques have been developed to cope with this problem but proven unwieldy to use and not amenable to formalization and property analysis due to their generality. Methods and techniques from two fields have been particularly useful in addressing such advanced modularization problems: Aspect-Oriented Software Development as the field concerned with the systematic handling of modularization issues and domain-specific languages that provide declarative and efficient means for the definition of crosscutting functionalities.

Aspect-Oriented Software Development [68], [41] has emerged over the previous decade as the domain of systematic exploration of crosscutting concerns and corresponding support throughout the software development process. The corresponding research efforts have resulted, in particular, in the recognition of *crosscutting* as a fundamental problem of virtually any large-scale application, and the definition and implementation of a large number of aspect-oriented models and languages.

However, most current aspect-oriented models, notably AspectJ [67], rely on pointcuts and advice defined in terms of individual execution events. These models are subject to serious limitations concerning the modularization of crosscutting functionalities in distributed applications, the integration of aspects with other modularization mechanisms such as components, and the provision of correctness guarantees of the resulting AO applications. They do, in particular, only permit the manipulation of distributed applications on a per-host basis, that is, without direct expression of coordination properties relating different distributed entities [86]. Similarly, current approaches for the integration of aspects and (distributed) components do not directly express interaction properties between sets of components but rather seemingly unrelated modifications to individual components [54]. Finally, current formalizations of such aspect models are formulated in terms of low-level semantic abstractions (see, e.g., Wand's et al semantics for AspectJ [89]) and provide only limited support for the analysis of fundamental aspect properties.

Different approaches have been put forward to tackle these problems, in particular, in the context of so-called *stateful* or *history-based aspect languages* [58], [59], which provide pointcut and advice languages that directly express rich relationships between execution events. Such languages have been proposed to directly express coordination and synchronization issues of distributed and concurrent applications [78], [48], [61], provide more concise formal semantics for aspects and enable analysis of their properties [44], [60], [58], [42]. Furthermore, first approaches for the definition of *aspects over protocols* have been proposed, as well as over regular structures [58] and non-regular ones [88], [76], which are helpful for the modular definition and verification of protocols over crosscutting functionalities.

They represent, however, only first results and many important questions concerning these fundamental issues remain open, in particular, concerning the semantics foundations of AOP and the analysis and enforcement of correctness properties governing its, potentially highly invasive, modifications.

Domain-specific languages (DSLs) represent domain knowledge in terms of suitable basic language constructs and their compositions at the language level. By trading generality for abstraction, they enable complex relationships among domain concepts to be expressed concisely and their properties to be expressed and formally analyzed. DSLs have been applied to a large number of domains; they have been particularly popular in the domain of software generation and maintenance [74], [92].

Many modularization techniques and tasks can be naturally expressed by DSLs that are either specialized with respect to the type of modularization constructs, such as a specific brand of software component, or to the compositions that are admissible in the context of an application domain that is targeted by a modular implementation. Moreover, software development and evolution processes can frequently be expressed by transformations between applications implemented using different DSLs that represent an implementation at different abstraction levels or different parts of one application.

Functionalities that crosscut a component-based application, however, complicate such a DSL-based transformational software development process. Since such functionalities belong to another domain than that captured by the components, different DSLs should be composed. Such compositions (including their syntactic expression, semantics and property analysis) have only very partially been explored until now. Furthermore, restricted composition languages and many aspect languages that only match execution events of a specific domain (e.g., specific file accesses in the case of security functionality) and trigger only domain-specific actions clearly are quite similar to DSLs but remain to be explored.

3.4. Distribution and Concurrency

While ASCOLA does not investigate distribution and concurrency as research domains per se (but rather from a software engineering and modularization viewpoint), there are several specific problems and corresponding approaches in these domains that are directly related to its core interests that include the structuring and modularization of large-scale distributed infrastructures and applications. These problems include crosscutting functionalities of distributed and concurrent systems, support for the evolution of distributed software systems, and correctness guarantees for the resulting software systems.

Underlying our interest in these domains is the well-known observation that large-scale distributed applications are subject to *numerous crosscutting functionalities* (such as the transactional behavior in enterprise information systems, the implementation of security policies, and fault recovery strategies). These functionalities are typically partially encapsulated in distributed infrastructures and partially handled in an ad hoc manner by using infrastructure services at the application level. Support for a more principled approach to the development and evolution of distributed software systems in the presence of crosscutting functionalities has been investigated in the field of *open adaptable middleware* [49], [71]. Open middleware design exploits the concept of reflection to provide the desired level of configurability and openness. However, these approaches are subject to several fundamental problems. One important problem is their insufficient, framework-based support that only allows partial modularization of crosscutting functionalities.

There has been some *criticism* on the use of *AspectJ-like aspect models* (which middleware aspect models like that of JBoss AOP are an instance of) for the modularization of distribution and concurrency related concerns, in particular, for transaction concerns [70] and the modularization of the distribution concern itself [86]. Both criticisms are essentially grounded in AspectJ's inability to explicitly represent sophisticated relationships between execution events in a distributed system: such aspects therefore cannot capture the semantic relationships that are essential for the corresponding concerns. History-based aspects, as those proposed by the ASCOLA project-team provide a starting point that is not subject to this problem.

From a point of view of language design and implementation, aspect languages, as well as domain specific languages for distributed and concurrent environments share many characteristics with existing distributed languages: for instance, event monitoring is fundamental for pointcut matching, different synchronization strategies and strategies for code mobility [64] may be used in actions triggered by pointcuts. However, these relationships have only been explored to a small degree. Similarly, the formal semantics and formal properties of aspect languages have not been studied yet for the distributed case and only rudimentarily for the concurrent one [44], [61].

3.5. Security

Security properties and policies over complex service-oriented and standalone applications become ever more important in the context of asynchronous and decentralized communicating systems. Furthermore, they constitute prime examples of crosscutting functionalities that can only be modularized in highly insufficient ways with existing programming language and service models. Security properties and related properties, such as accountability properties, are therefore very frequently awkward to express and difficult to analyze and enforce (provided they can be made explicit in the first place).

Two main issues in this space are particularly problematic from a compositional point of view. First, information flow properties of programming languages, such as flow properties of Javascript [46], and service-based systems [52] are typically specially-tailored to specific properties, as well as difficult to express and analyze. Second, the enforcement of security properties and security policies, especially accountability-related properties [80], [87], is only supported using ad hoc means with rudimentary support for property verification.

The ASCOLA team has recently started to work on providing formal methods, language support and implementation techniques for the modular definition and implementation of information flow properties as well as policy enforcement in service-oriented systems as well as, mostly object-oriented, programming languages.

3.6. Green IT

With the emergence of the Future Internet and the dawn of new IT architecture and computation models such as cloud computing, the usage of data centers (DC) as well as their power consumption increase dramatically [51]. Besides the ecological impact [65], energy consumption is a predominant criterion for DC providers since it determines the daily cost of their infrastructure. As a consequence, power management becomes one of the main challenges for DC infrastructures and more generally for large-scale distributed systems.

To address this problem, we study two approaches: a workload-driven [47] and power-driven one [83]. As part of the workload-driven solution, we adapt the power consumption of the DC depending on the application workload, and evaluate whether this workload to be more reactive. We develop a distributed system from the system to the service-oriented level mainly based on hardware and virtualization capabilities that is managed in a user-transparent fashion. As part of the power-driven approach, we address energy consumption issues through a strong synergy inside the infrastructure software stack and more precisely between applications and resource management systems. This approach is characterized by adapting QoS properties aiming at the best trade-off between cost of energy (typically from the regular electric grid), its availability (for instance, from renewable energy), and service degradation caused, for instance, by application reconfigurations to jobs suspensions.

3.7. Capacity Planning for Large Scale Distributed System

Since the last decade, cloud computing has emerged as both a new economic model for software (provision) and as flexible tools for the management of computing capacity [45]. Nowadays, the major cloud features have become part of the mainstream (virtualization, storage and software image management) and the big market players offer effective cloud-based solutions for resource pooling. It is now possible to deploy virtual infrastructures that involve virtual machines (VMs), middleware, applications, and networks in such a simple manner that a new problem has emerged since 2010: VM sprawl (virtual machine proliferation) that consumes valuable computing, memory, storage and energy resources, thus menacing serious resource shortages. Scientific approaches that address VM sprawl are both based on classical administration techniques like the lifecycle management of a large number of VMs as well as the arbitration and the careful management of all resources consumed and provided by the hosting infrastructure (energy, power, computing, memory, network etc.) [63], [91].

The ASCOLA team investigates fundamental techniques for cloud computing and capacity planning, from infrastructures to the application level. Capacity planning is the process of planning for, analyzing, sizing, managing and optimizing capacity to satisfy demand in a timely manner and at a reasonable cost. Applied to distributed systems like clouds, a capacity planning solution must mainly provide the minimal set of resources necessary for the proper execution of the applications (i.e., to ensure SLA). The main challenges in this context are: scalability, fault tolerance and reactivity of the solution in a large-scale distributed system, the analysis and optimization of resources to minimize the cost (mainly costs related to the energy consumption of datacenters), as well as the profiling and adaptation of applications to ensure useful levels of quality of service (throughput, response time, availability etc.).

Our solutions are mainly based on virtualized infrastructures that we apply from the IaaS to the SaaS levels. We are mainly concerned by the management and the execution of the applications by harnessing virtualization capabilities, the investigation of alternative solutions that aim at optimizing the trade-off between performance and energy costs of both applications and cloud resources, as well as arbitration policies in the cloud in the presence of energy-constrained resources.

4. Application Domains

4.1. Enterprise Information Systems and Services

Large IT infrastructures typically evolve by adding new third-party or internally-developed components, but also frequently by integrating already existing information systems. Integration frequently requires the addition of glue code that mediates between different software components and infrastructures but may also consist in more invasive modifications to implementations, in particular to implement crosscutting functionalities. In more abstract terms, enterprise information systems are subject to structuring problems involving horizontal composition (composition of top-level functionalities) as well as vertical composition (reuse and sharing of implementations among several top-level functionalities). Moreover, information systems have to be more and more dynamic.

Service-Oriented Computing (SOC) that is frequently used for solving some of the integration problems discussed above. Indeed, service-oriented computing has two main advantages:

- Loose-coupling: services are autonomous: they do not require other services to be executed;
- Ease of integration: Services communicate over standard protocols.

Our current work is based on the following observation: similar to other compositional structuring mechanisms, SOAs are subject to the problem of crosscutting functionalities, that is, functionalities that are scattered and tangled over large parts of the architecture and the underlying implementation. Security functionalities, such as access control and monitoring for intrusion detection, are a prime example of such a functionality in that it is not possible to modularize security issues in a well-separated module. Aspect-Oriented Software Development is precisely an application-structuring method that addresses in a systemic way the problem of the lack of modularization facilities for crosscutting functionalities.

We are considering solutions to secure SOAs by providing an aspect-oriented structuring and programming model that allows security functionalities to be modularized. Two levels of research have been identified:

- Service level: as services can be composed to build processes, aspect weaving will deal with the orchestration and the choreography of services.
- Implementation level: as services are abstractly specified, aspect weaving will require to extend service interfaces in order to describe the effects of the executed services on the sensitive resources they control.

In 2015, we have published results on constructive mechanisms for security and accountability properties in service-based systems as well as results on service provisioning problems, in particular, service interoperability and mediation. Furthermore, we take part in the European project A4Cloud on accountability challenges, that is, the responsible stewardship of third-party data and computations, see Sec. 9.3.

4.2. Capacity Planning in Cloud, Fog and Edge Computing

Cloud and more recently Fog and Edge computing platforms aim at delivering large capacities of computing power. These capacities can be used to improve performance (for scientific applications) or availability (e.g., for Internet services hosted by datacenters). These distributed infrastructures consist of a group of coupled computers that work together and may be spread across a LAN (cluster), across a the Internet (Fog/Edge). Due to their large scale, these architectures require permanent adaptation, from the application to the system level and call for automation of the corresponding adaptation processes. We focus on self-configuration and self-optimization functionalities across the whole software stack: from the lower levels (systems mechanisms such as distributed file systems for instance) to the higher ones (i.e. the applications themselves such as clustered servers or scientific applications).

In 2015, we have proposed VMPlaces, a dedicated framework to evaluate and compare VM placement algorithms. Globally the framework is composed of two major components: the injector and the VM placement algorithm. The injector constitutes the generic part of the framework (i.e. the one you can directly use) while the VM placement algorithm is the component a user wants to study (or compare with other existing algorithms), see Sec. 7.2.

In the energy field, we have designed a set of techniques, named Optiplace, for cloud management with flexible power models through constraint programming. OptiPlace supports external models, named views. Specifically, we have developed a power view, based on generic server models, to define and reduce the power consumption of a datacenter's physical servers. We have shown that OptiPlace behaves at least as good as our previous system, Entropy, requiring as low as half the time to find a solution for the constrained-based placement of tasks for large datacenters.

4.3. Pervasive Systems

Pervasive systems are another class of systems raising interesting challenges in terms of software structuring. Such systems are highly concurrent and distributed. Moreover, they assume a high-level of mobility and context-aware interactions between numerous and heterogeneous devices (laptops, PDAs, smartphones, cameras, electronic appliances...). Programming such systems requires proper support for handling various interfering concerns like software customization and evolution, security, privacy, context-awareness... Additionally, service composition occurs spontaneously at runtime.

Like Pervasive systems, Internet of Things is a major theme of these last ten years. Many research works has been led on the whole chain, from communicating sensors to big data management, through communication middlewares. Few of these works have addressed the problem of gathered data access.

The more a sensor networks senses various data, the more the users panel is heterogeneous. Such an heterogeneity leads to a major problem about data modeling: for each user, to aim at precisely addressing his needs and his needs only; ie to avoid a data representation which would overwhelm the user with all the data sensed from the network, regardless if he needs it or not. To leverage this issue, we have proposed a multitree modeling for sensor networks which addresses each of these specific usages.With this modeling comes a domain specific language (DSL) which allows users to manipulate, parse and aggregate information from the sensors.

In 2014, we have extended the language EScala, which integrates reactive programming through events with aspect-oriented and object-oriented mechanisms.

5. Highlights of the Year

5.1. Highlights of the Year

This year the team has produced major results in the domains of the foundations of computer science as well as capacity management for large-scale distributed software systems.

Concerning the foundations of computer science, we have presented new results on the provably correct execution of programs that are only partially typed [22] and generalized the use of dependent types with side effects [26].

As to distributed systems, we have introduced a new cloud model that provides QoS-levels and SLA as first-class citizens of cloud-based systems [19]. Furthermore, we have provided new mechanisms for the privacy-preserving storage of data of a user over clouds managed by different cloud providers [30].

6. New Software and Platforms

6.1. CSLA

Cloud Service Level Agreement language

KEYWORDS: Cloud computing - Service-level agreement - Elasticity
FUNCTIONAL DESCRIPTION

CSLA, the Cloud Service Level Agreement language, allows the definition of SLA properties for arbitrary Cloud services (XaaS). CSLA addresses QoS uncertainty in unpredictable and dynamic environment and provides a cost model of Cloud computing. Besides the standard formal definition of contracts – comprising validity, parties, services definition and guarantees/violations – CSLA is enriched with features, such as QoS degradation and an advanced penalty model, thus introducing fine-grained language support for Cloud elasticity management.

- Participants: Thomas Ledoux and Yousri Kouki
- Contact: Thomas Ledoux
- URL: http://www.emn.fr/z-info/csla/

6.2. CSQL

Cryptographic Composition for Query Language

SCIENTIFIC DESCRIPTION

C2QL is a compositional language of security techniques for information privacy in the cloud. A cloud service can use security techniques to ensure information privacy. These techniques protect privacy by converting the client's personal data into unintelligible text. But they also cause the loss of some functionalities of the service. As a solution, CSQL permits to compose security techniques to ensure information privacy without the loss of functionalities. But, the composition makes the writing of programs more intricate. To help the programmer, C2QL defines a query language for the definition of cloud services that enforces information privacy with the composition of security techniques. This language comes with a set of algebraic laws to, systematically, transform a local service without protection into its cloud equivalent that is protected by composition.

FUNCTIONAL DESCRIPTION

C2QL is implemented in Idris, a functional language of the Haskell family. The implementation harnesses the Idris dependant type system to ensure the correct composition of security mechanisms and provides a transformation of the implementation into a π-calculus. This transformation serves two purposes. First, it makes the distribution explicit, showing how a computation is distributed over SaaS, PaaS and client applications. Then, it helps defining an encoding into ProVerif to check that the service preserves the privacy of its clients.

- Participants: Ronan-Alexandre Cherrueau, Rémi Douence, Mario Südholt
- Contact: Ronan-Alexandre Cherrueau
- URL: https://github.com/rcherrueau/C2QL

6.3. EScala

SCIENTIFIC DESCRIPTION

EScala extends the idea of events as object members, as realized by C# events, with the possibility to define events declaratively by expressions over other events. The occurrences of an event can be defined by various set operations, such as union, intersection and difference, applied on the occurrences of other events. Events can be filtered by arbitrary conditions, the data attached to the events can be transformed by arbitrary functions. Event expressions make it possible to define events in terms of other events, at the lowest level relying on primitive events.
FUNCTIONAL DESCRIPTION

EScala is an extension of Scala programming language with support for events as attributes of objects. The support for events in EScala, combine the ideas of event-driven, aspect-oriented and functional-reactive programming.

- Participants: Jacques Noyé and Jurgen Van Ham
- Contact: Jurgen Van Ham
- URL: http://www.stg.tu-darmstadt.de/research/escala/index.en.jsp

6.4. JEScala

FUNCTIONAL DESCRIPTION

JEScala extends EScala with support for concurrent programming. Events can be declared as asynchronous so that their handling takes place concurrently. A new composition operator, the join operator, inspired by the join calculus, can also be used to synchronize the concurrent activities created by asynchronous events and communicate between them.

- Participants: Jurgen Van Ham and Jacques Noyé
- Contact: Jurgen Van Ham
- URL: http://www.stg.tu-darmstadt.de/research/jescala_menu/index.en.jsp

6.5. SimGrid

Scientific Instrument for the study of Large-Scale Distributed Systems
KEYWORDS: Large-scale Emulators - Grid Computing - Distributed Applications
FUNCTIONAL DESCRIPTION

SimGrid is a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. The simulation engine uses algorithmic and implementation techniques toward the fast simulation of large systems on a single machine. The models are theoretically grounded and experimentally validated. The results are reproducible, enabling better scientific practices.

Its models of networks, CPUs and disks are adapted to (Data)Grids, P2P, Clouds, Clusters and HPC, allowing multi-domain studies. It can be used either to simulate algorithms and prototypes of applications, or to emulate real MPI applications through the virtualization of their communication, or to formally assess algorithms and applications that can run in the framework.

The formal verification module explores all possible message interleavings in the application, searching for states violating the provided properties. We recently added the ability to assess liveness properties over arbitrary and legacy codes, thanks to a system-level introspection tool that provides a finely detailed view of the running application to the model checker. This can for example be leveraged to verify both safety or liveness properties, on arbitrary MPI code written in C/C++/Fortran.

- Participants: Frederic Suter, Martin Quinson, Arnaud Legrand, Takahiro Hirofuchi, Adrien Lebre, Jonathan Pastor, Mario Sudholt, Luka Stanisic, Augustin Degomme, Jean Marc Vincent, Florence Perronnin and Jonathan Rouzaud-Cornabas
- Partners: CNRS - ENS Rennes - Université de Nancy
- Contact: Martin Quinson
- URL: http://simgrid.gforge.inria.fr/

6.6. VMPlaces

FUNCTIONAL DESCRIPTION

VMPlaces is a dedicated framework to evaluate and compare VM placement algorithms. This framework is composed of two major components: the injector and the VM placement algorithm. The injector is the generic part of the framework (i.e. the one you can directly use) while the VM placement algorithm is the part you want to study (or compare with available algorithms). Currently, the VMPlaceS is released with three algorithms:

Entropy, a centralized approach using a constraint programming approach to solve the placement/reconfiguration VM problem

Snooze, a hierarchical approach where each manager of a group invokes Entropy to solve the placement/reconfiguration VM problem. Note that in the original implementation of Snooze, it is using a specific heuristic to solve the placement/reconfiguration VM problem. As the sake of simplicity, we have simply reused the entropy scheduling code.

DVMS, a distributed approach that dynamically partitions the system and invokes Entropy on each partition.

- Participants: Takahiro Hirofuchi, Adrien Lebre, Jonathan Pastor, Flavien Quesnel and Mario Sudholt
- Contact: Adrien Lebre
- URL: http://beyondtheclouds.github.io/VMPlaceS/

6.7. btrCloud

KEYWORDS: Cloud computing - Virtualization - Grid - Energy - Orchestration - Autonomic system - Placement - Cluster - Data center - Scheduler
FUNCTIONAL DESCRIPTION

Orchestration, virtualization, energy, autonomic system, placement, cloud computing, cluster, data center, scheduler, grid

btrCloud is a virtual machine manager for clusters and provides a complete solution for the management and optimization of virtualized data centers. btrCloud (acronym of better cloud) is composed of three parts.

The analysis function enables operatives and people in charge to monitor and analyze how a data-center works - be it on a daily basis, on the long run, or in order to predict future trends. This feature includes boards for performance evaluation and analysis as well as trends estimation.

btrCloud, by the integration of btrScript, provides (semi-)automated VM lifecycle management, including provisioning, resource pool management, VM tracking, cost accounting, and scheduled deprovisioning. Key features include a thin client interface, template-based provisioning, approval workflows, and policy-based VM placement.

Finally, several kinds of optimizations are currently available, such as energy and load balancing. The former can help save up to around 20% of the data-center energy consumption. The latter provides optimized quality of service properties for applications that are hosted in the virtualized datacenters.

- Participants: Guillaume Le Louet, Frederic Dumont and Jean-Marc Menaud
- Contact: Jean-Marc Menaud
- URL: http://www.btrcloud.org/btrCloud/index_EN.html

7. New Results

7.1. Software composition and programming languages

Participants: Walid Benghrabit, Ronan-Alexandre Cherrueau, Rémi Douence, Hervé Grall, Florent Marchand de Kerchove de Denterghem, Jacques Noyé, Jean-Claude Royer, Mario Südholt.

This year we have published a number of new results in the domains of software composition and programming languages that range from pragmatic ones like modularity issues to formal studies in the domain of dependent type theory via static analysis and formal verification.

7.1.1. Formal Methods, logics and type theory

Concerning verification and formal semantics, we have defined the semantics of our dependent interoperability framework and we propose the notion the partial type equivalences as a key feature. We have also studied proofs in dependent type theory and synthesized call-by-value and call-by-name translations.

7.1.1.1. Verified Dependent Interoperability.

Full-spectrum dependent types promise to enable the development of correct-by-construction software. However, even certified software needs to interact with simply-typed or untyped programs, be it to perform system calls, or to use legacy libraries. Trading static guarantees for runtime checks, the dependent interoperability framework provides a mechanism by which simply-typed values can safely be coerced to dependent types and, conversely, dependently-typed programs can defensively be exported to a simply-typed application. In [22], we give a semantic account of dependent interoperability. Our presentation relies on and is guided by a pervading notion of type equivalence, whose importance has been emphasized in recent works on homotopy type theory. Specifically, we develop the notion of partial type equivalences as a key foundation for dependent interoperability. Our framework is developed in Coq; it is thus constructive and verified in the strictest sense of the terms. Using our library, users can specify domain-specific partial equivalences between data structures. Our library then takes care of the (sometimes, heavy) lifting that leads to interoperable programs. It thus becomes possible, as we shall illustrate, to internalize and hand-tune the extraction of dependently-typed programs to interoperable OCaml programs within Coq itself.

7.1.1.2. Forcing in Type Theory.

In [26], we study forcing translations of proofs in dependent type theory, through the Curry-Howard correspondence. Based on a call-by-push-value decomposition, we synthesize two simply-typed translations: i) one call-by-value, corresponding to the translation derived from the presheaf construction as studied in a previous paper; ii) one call-by-name, whose intuitions already appear in Krivine and Miquel's work. Focusing on the call-by-name translation, we adapt it to the dependent case and prove that it is compatible with the definitional equality of our system, thus avoiding coherence problems. This allows us to use any category as forcing conditions, which is out of reach with the call-by-value translation. Our construction also exploits the notion of storage operators in order to interpret dependent elimination for inductive types. This is a novel example of a dependent theory with side-effects, clarifying how dependent elimination for inductive types must be restricted in a non-pure setting. Being implemented as a Coq plugin, this work gives the possibility to formalize easily consistency results, for instance the consistency of the negation of Voevodsky's univalence axiom.

7.1.2. Programming languages

In the domain of programming languages we have presented new results on constraint programming, development of correct programs by construction and better controls for computational effects and modularity for JavaScript.

7.1.2.1. Constraint programming

Constraint programming (CP) relies on filtering algorithms in order to deal with combinatorial problems. Global constraints offer efficient algorithms for complex constraints. In particular a large family of global constraints can be expressed as constraints of finite state automata with counters. We have generalized these automata constraints in order to compose them as transducers [16]. We have also extended these results with different techniques [20]. First, we have improved the automaton synthesis to generate automata with fewer accumulators. Second, we have shown how to decompose a constraint specified by an automaton with accumulators into a conjunction of linear inequalities, for use by a MIP (Mixed-Integer Programming) solver. Third, we have generalized the implied constraint generation to cover the entire family of time-series constraints. The newly synthesized automata for time-series constraints outperform the old ones, for both the CP and MIP decompositions, and the generated implied constraints boost the inference, again for both the CP and MIP decompositions.

7.1.2.2. Program correctness

Most IDEs provide refactoring tools to assist programmers when they modify the structure of their software. However the refactoring facilities of many popular tools (Eclipse, Visual Studio, IntelliJ, etc.) are currently not reliable : they occasionally change the program semantics in unexpected ways, and, as a result, the programmers systematically have to re-test the resulting code. We have build a refactoring tool for C programs which core operation is proved correct by construction [21]. To do that, we build an AST transformation with Coq (based on the CompCert C implementation) and we prove that this transformation preserves the external behavior of programs. The code of the transformation is then extracted to OCaml and is then embedded in a traditional parse/transform/pretty-print setting to provide a working prototype.

7.1.2.3. Effect Capabilities

Computational effects complicate the tasks of reasoning about and maintaining software, due to the many kinds of interferences that can occur. While different proposals have been formulated to alleviate the fragility and burden of dealing with specific effects, such as state or exceptions, there is no prevalent robust mechanism that addresses the general interference issue. Building upon the idea of capability-based security, we propose in [18] effect capabilities as an effective and flexible manner to control monadic effects and their interferences. Capabilities can be selectively shared between modules to establish secure effect-centric coordination. We further refine capabilities with type-based permission lattices to allow fine-grained decomposition of authority. We provide an implementation of effect capabilities in Haskell, using type classes to establish a way to statically share capabilities between modules, as well as to check proper access permissions to effects at compile time. We first exemplify how to tame effect interferences using effect capabilities by treating state and exceptions. Then we focus on taming I/O by proposing a fine-grained lattice of I/O permissions based on the current classification of its operations. Finally, we show that integrating effect capabilities with modern tag-based monadic mechanisms provides a practical, modular and safe mechanism for monadic programming in Haskell.

7.1.2.4. Extensible JavaScript Modules

As part of the SecCloud project, we have studied how to modularly extend JavaScript interpreters with dynamic security analyses in particular information flow analyses. This has led us to study ways to improve on the standard JavaScript module pattern. This pattern is commonly used to encapsulate definitions by using closures. However, closures prevent module definitions from being extended at runtime. We have proposed a simple pattern that not only opens the module, but allows one to extend the module definitions in layers [39]. The pattern leverages the with construct and the prototype delegation mechanism of JavaScript to mimic a form of dynamic binding, while minimizing the changes made to the module code.

Florent Marchand's PhD thesis [13] details the proposal further and shows its application to the modular extension of Narcissus, a full-blown JavaScript interpreter, with several dynamic analyses, including the information flow of Austin and Flanagan based on multiple facets. A comparison with a previous ad hoc implementation of the analysis illustrates the benefits of the proposal.

7.1.3. Software Security and Privacy

In the area of security we have focused on expressing advanced security concerns with abstract and formal languages and the study of policy monitoring and the detection of conflicts.

7.1.3.1. Runtime verification of advanced logical security properties.

Monitoring or runtime verification means to observe the system execution and to check if it deviates or not from a predefined contract. Our contract is a formula written in AAL (Abstract Accountability Language) expressing the expected behavior of a system, the audit steps as well as punishment and compensation. We choose to use the rewriting approach with the three valued logic as many other existing approaches. The monitoring problem raised a validity question, if we start with a formula neither true nor false are we sure to conclude? The response is no and this is a completeness problem and all published solutions are incomplete. For LTL, mixing the standard semantics, the rewriting principle and coinduction we are able to define a complete monitoring mechanism. A first implementation has been done into our AccLab tool support and

sketched in [38]. We are investigating the extension of our LTL rewriting mechanism to cope with the first-order case.

7.1.3.2. Specification of advanced security and privacy properties.

Security and privacy requirements in ubiquitous systems need a sophisticated policy language with features to express access restrictions and obligations. Ubiquitous systems involve multiple actors owning sensitive data concerning aspects such as location, discrete and continuous time, multiple roles that can be shared among actors or evolve over time. Conflict management is an important problem in security policy frameworks. In [31] we present an abstract language (AAL) dedicated to accountability. We show how to specify most of these security and privacy features and compare it with the XACML approach. We also classified the existing conflict detection for XACML like approaches in dynamic, testing, or static detection. A thorough analysis of these mechanisms reveals that they have several weaknesses and they are not applicable in our context. We advocate for a classic approach using the notion of logical consistency to detect conflicts in AAL.

7.1.3.3. Composition of privacy-enhancing and security mechanisms.

As part of his PhD thesis [11], Ronan Cherrueau's has defined a language for the composition of three privacy-enhancing and security mechanisms: symmetric key encryption, database fragmentation and on-client computations. The language allows the expression of distributed programs that protect data by applying compositions of the three mechanisms to them. The language ensures basic privacy and security properties by a type system based on dependent types. This type system ensures, for example, that data that has been encrypted and stored in a database fragment cannot be accessed in plain form and from another location than that fragment. Furthermore, the language comes equiped with four major additional results. First, a calculus that allows for the semi-automatic derivation of distributed privacy-preserving and secure programs from an original non-distributed one. Second, a transformation from the language to the π-calculus. Third, a transformation into an input specification to the Proverif model checker for security properties. Fourth, two implementations on the basis of, respectively, the Scala and Idris languages that harness their corresponding dependent type systems.

7.2. Distributed programming and the Cloud

Participants: Frederico Alvares, Bastien Confais, Simon Dupont, Md Sabbir Hasan, Adrien Lebre, Thomas Ledoux, Guillaume Le Louët, Jean-Marc Menaud, Jonathan Pastor, Rémy Pottier, Anthony Simonet, Mario Südholt.

7.2.1. Cloud applications and infrastructures

Complex event processing. We presented this year the evolution of SensorScript towards a language for complex event processing dedicated to sensor networks. While the model mainly relies on previous works, we highlighted how the new language builds on the multitree in order to provide complex event processing mechanisms. We are able to balance the syntactic concision of the language with a real-time complex event processor for sensor networks. By providing flexible selections over the nodes, with the possibility to filter them on complex conditions, possibly over a time window, we offer a strong alternative to traditional SQL used in the literature. Moreover, SensorScript does not focus only on data access. In fact it provides the possibility to widen the scope of the methods accessible on nodes to other features than sensors monitoring, including but not limited to addressing actuators functions. Finally we showed that SensorScript is able to address examples proposed in the literature, with simpler results than SQL, while highlighting its limitations, especially on history management. [24]

Secure cloud storage. The increasing number of cloud storage services like Dropbox or Google Drive allows users to store more and more data on the Internet. However, these services do not give users enough guarantees in protecting the privacy of their data. In order to limit the risk that the storage service scans user documents for commercial purposes, we propose a storage service that stores data on several cloud providers while preventing these providers to read user documents. TrustyDrive is a cloud storage service that protects the privacy of users by breaking user documents into blocks in order to spread them on several cloud providers. As cloud providers

only own a part of the blocks and they do not know the block organization, they can not read user documents. Moreover, the storage service connects directly users and cloud providers without using a third-party as is generally the practice in cloud storage services. Consequently, users do not give critical information (security keys, passwords, etc.) to a third-party. [30]

7.2.1.1. Service-level agreement for the Cloud.

Quality-of-service and SLA guarantees are among the major challenges of cloud-based services. In [19], we first present a new cloud model called SLAaaS — SLA aware Service. SLAaaS considers QoS levels and SLA as first class citizens of cloud-based services. This model is orthogonal to other SaaS, PaaS, and IaaS cloud models, and may apply to any of them. More specifically, we make three contributions: (i) we provide a domain-specific language that allows to define SLA constraints in cloud services; (ii) we present a general control-theoretic approach for managing cloud service SLA; (iii) we apply our approach to MapReduce, locking, and e-commerce services.

7.2.1.2. Cloud Capacity Planning and Elasticity.

Capacity management is a process used to manage the capacity of IT services and the IT infrastructure. Its primary goal is to ensure that IT resources (services, infrastructure) are right-sized to meet current and future requirements in a cost-effective and timely manner. In [34], we present a comprehensive overview of capacity planning and management for cloud computing. First, we state the problem of capacity management in the context of cloud computing from the point of view of several service providers. Second, we provide a brief discussion about *when* capacity planning should take place. Finally, we survey a number of methods for capacity planning and management proposed by both people from industry and researchers.

In his PhD [12], Simon Dupont proposes to extend the concept of elasticity to higher layers of the cloud, and more precisely to the SaaS level. He presents the new concept of *software elasticity* by defining the ability of the software to adapt, ideally in an autonomous way, to cope with workload changes and/or limitations of IaaS elasticity. This brings the consideration of Cloud elasticity in a multi-layer way through the adaptation of all kind of Cloud resources (software, virtual machines, physical machines). In [23], we introduce ElaScript, a DSL that offers Cloud administrators a simple and concise way to define complex elasticity-based reconfiguration plans. ElaScript is capable of dealing with both infrastructure and software elasticities, independently or together, in a coordinated way. We validate our approach by first showing the interest to have a DSL offering multiple levels of control for Cloud elasticity, and then by showing its integration with a realistic well-known application benchmark deployed in OpenStack and Grid'5000 infrastructure testbed.

7.2.1.3. Infrastructure.

Academic and industry experts are now advocating for going from large-centralized Cloud Computing infrastructures to smaller ones massively distributed at the edge of the network (aka., Fog and Edge Computing solutions). Among the obstacles to the adoption of this model is the development of a convenient and powerful IaaS system capable of managing a significant number of remote data-centers in a unified way.

In 2016, we achieved three major results in this context.

The first result is related to the economical viability of Fog/Edge Computing infrastructures that is often debated w-r-t large cloud computing data centers operated by US giants such as Amazon, Google .... To answer such a question, we conducted a specific study that goes beyond the state of the art of the current cost model of Distributed Cloud infrastructures. First, we provided a classification of the different ways of deploying Distributed Cloud platforms. Then, we proposed a versatile cost model that can help new actors evaluate the viability of deploying a Fog/Edge Computing offer. We illustrated the relevance of our proposal by instantiating it over three use-cases and comparing them according to similar computation capabilities provided by the Amazon solution. Such a study clearly showed that deploying a Distributed Cloud infrastructure makes sense for telcos as well as new actors willing to enter the game [29].

The second result is related to the preliminary revisions we made in OpenStack. The OpenStack software suite has become the de facto open-source solution to operate, supervise and use a Cloud Computing infrastructure. Our objective is to study to what extent current OpenStack mechanisms can handle massively distributed

cloud infrastructures and to propose revisions/extensions of internal mechanisms when appropriate. The work we conducted this year focused on the Nova service of OpenStack.More precisely, we modified the code base in order to use a distributed key/value store instead of the centralized SQL backend. We conducted several experiments that validate the correct behavior and gives performance trends of our prototype through an emulation of several data-centers using Grid'5000 testbed. In addition to paving the way to the first large-scale and Internet-wide IaaS manager, we expect this work will attract a community of specialists from both distributed system and network areas to address the Fog/Edge Computing challenges within the OpenStack ecosystem [36], [27]. These and additional corresponding results have been presented in a more detailed manner as part of Jonathan Pastor's PhD thesis [14].

The third result is related to the data management in Fog/Edge Computing infrastructures. Our ultimate goal is to propose an Amazon-S3 like system, *i.e.*, a blob storage service, that can take into account Fog/Edge specifics. The study we achieved this year is preliminary. We first identified a list of properties a storage system should meet in this context. Second, we evaluated through performance analysis three "off-the-shelf" object store solutions, namely Rados, Cassandra and InterPlanetary File System (IPFS). In particular, we focused (i) on access times to push and get objects under different scenarios and (ii) on the amount of network traffic that is exchanged between the different sites during such operations. We also evaluated how the network latencies influence the access times and how the systems behave in case of network partitioning. Experiments have been conducted using the Yahoo Cloud System Benchmark (YCSB) on top of the Grid'5000 testbed. We showed that among the three tested solutions IPFS fills most of the criteria expected for a Fog/Edge computing infrastructure. [33], [32]

7.2.2. Renewable energy

With the emergence of the Future Internet and the dawning of new IT models such as cloud computing, the usage of data centers (DC), and consequently their power consumption, increase dramatically. Besides the ecological impact, the energy consumption is a predominant criterion for DC providers since it determines the daily cost of their infrastructure. As a consequence, power management becomes one of the main challenges for DC infrastructures and more generally for large-scale distributed systems. We have design the EpoCloud prototype, from hardware to middleware layers. This prototype aims at optimizing the energy consumption of mono-site Cloud DCs connected to the regular electrical grid and to renewable-energy sources. [17]

7.2.2.1. Green Energy awareness in SaaS Application.

With the proliferation of Cloud computing, data centers have to urgently face energy consumption issues. Although recent efforts such as the integration of renewable energy to data centers or energy efficient techniques in (virtual) machines contribute to the reduction of carbon footprint, creating green energy awareness around *Interactive Cloud Applications* by smartly using the presence of green energy has not been yet addressed. By *awareness*, we mean the inherited capability of SaaS applications to dynamically adapt with the availability of green energy and to reduce energy consumption while green energy is scarce or absent. In [25], we present two application controllers based on different metrics (e.g., availability of green energy, response time, user experience level). Based on extensive experiments with a real application benchmark and workloads in Grid'5000, results suggest that providers revenue can be increased as high as 64%, while 13% brown energy can be reduced without deprovisioning any physical or virtual resources at IaaS layer and 17 fold increment of performance can be guaranteed.

8. Bilateral Contracts and Grants with Industry

8.1. Cooperation with SIGMA group

Participants: Thomas Ledoux [correspondent], Simon Dupont.

In 2012, we have started a cooperation with Sigma Group (http://www.sigma.fr), a software editor and consulting enterprise. The cooperation consists in a joint (a so-called Cifre) PhD on eco-elasticity of software for the Cloud and the sponsorship of several engineering students at the MSc-level.

As a direct consequence of the increasing popularity of Cloud computing solutions, data centers are rapidly growing in number and size and have to urgently face with energy consumption issues. The aim of Simon Dupont's PhD, started in November 2012, is to explore the *software elasticity* capability in Software-as-a-Service (SaaS) development to promote the management of SaaS applications that are more flexible, more reactive to environment changes and therefore self-adaptive for a wider range of contexts. As a result, SaaS applications become more elastic and by transitivity more susceptible to energy constraints and optimization issues.

In 2016, Simon Dupont defended his PhD on "Cross-layer elasticity management for Cloud: towards an efficient usage of Cloud resources and services" [12]. Besides, we focused on ElaScript, a domain-specific language that offers Cloud administrators a simple and concise way to define complex elasticity-based reconfiguration plans [23].

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. RFI Atlanstic 2020

9.1.1.1. CoMe4ACloud

Participants: Thomas Ledoux [coordinator], Frederico Alvares, Zakarea Al Shara.

The high-level objective of the 1-year CoMe4ACloud (Constraints and Model Engineering for Autonomic Clouds) project is to provide an end-to-end solution for autonomic Cloud services. To that end, we rely on techniques of Constraint Programming so as a decision-making tool and Model-driven Engineering to ease the automatic generation of the so-called autonomic managers as well as their communication with the managed system.

CoMe4ACloud is an Atlanstic2020 funded project and supports a post-doc position. The project is led by Ascola research team and involves also AtlanModels and TASC, all of them from the LINA (Nantes Computer Science Laboratory) and situated at Ecole des Mines de Nantes. See https://come4acloud.github.io for more information.

9.1.2. Pays de la Loire

9.1.2.1. SyMeTRIC

Participant: Jean-Marc Menaud.

SyMeTRIC is a regional federated project in Systems Medicine funded by the Pays de la Loire region. Systems Medicine approaches can be compared to Systems Biology. They aim at integrating several information sources to design and validate bio-models and biomarkers to anticipate and enhance patients following (diagnosis, treatment response prediction, prognosis).

The long term goal of SyMeTRIC is to build a common Systems Medicine computing infrastructure to accelerate the discovery and validation of biomarkers in the fields of oncology, transplantation, and chronic cardiovascular diseases.

9.2. National Initiatives

9.2.1. CominLabs laboratory of excellence

9.2.1.1. EPOC

Participants: Jean-Marc Menaud [coordinator], Thomas Ledoux, Md Sabbir Hasan, Yunbo Li.

The project EPOC (Energy Proportional and Opportunistic Computing system) is a project running for 4 years. Four other partners collaborate within the project that is coordinated by ASCOLA: Myriads team, and the three institutions ENIB, ENSTB and University of Nantes. In this project, the partners focus on energy-aware task execution from the hardware to application's components in the context of a *mono-site* data center (all resources are in the same physical location) which is connected to the *regular electric Grid and to renewable energy sources* (such as windmills or solar cells). Three major challenges are addressed in this context: Optimize the energy consumption of distributed infrastructures and service compositions in the presence of ever more dynamic service applications and ever more stringent availability requirements for services; Design a clever cloud's resource management which takes advantage of renewable energy availability to perform opportunistic tasks, then exploring the trade-off between energy saving and performance aspects in large-scale distributed system; Investigate energy-aware optical ultra high-speed interconnection networks to exchange large volumes of data (VM memory and storage) over very short periods of time.

One of the strengths of the project is to provide a systematic approach, and use a single model for the system (from hard to soft) by mixing constraint programming and behavioral models to manage energy consumption in data centers.

9.2.1.2. PrivGen

Participants: Fatima-Zahra Boujdad, Mario Südholt [coordinator].

PrivGen ("Privacy-preserving sharing and processing of genetic data") is a three-year project that has been started in Oct. 2016 and is conducted by three partners: a team of computer scientists from the LATIM Inserm institute in Brest mainly working on data watermarking techniques, a team of geneticians from an Inserm institute in Rennes working on the gathering and interpretation of genetic data, and the Ascola team. The project provides funding of 330 KEUR altogether with an Ascola share of 120 KEUR.

The project considers challenges related to the outsourcing of genetic data that is in the Cloud by different stakeholders (researchers, organizations, providers, etc.). It tackles several limitations of current security solutions in the cloud, notably the lack of support for different security and privacy properties at once and computations executed at different sites that are executed on behalf of multiple stakeholders.

The partners are working on three main challenges:

- Mechanisms for a continuous digital content protection
- Composition of security and privacy-protection mechanisms
- Distributed processing and sharing of genetic data

The Ascola team is mainly involved in providing solutions for the second and third challenges.

9.2.1.3. SecCloud

Participants: Jacques Noyé [coordinator], Florent Marchand de Kerchove de Denterghem, Mario Südholt.

The high-level objective of the 3-year SecCloud (Secure Scripting for the Cloud) project is to enhance the security of devices on which web applications can be downloaded, i.e. to enhance client-side security in the context of the Cloud. In order to do so, the project relies on a language-based approach, focusing on three related issues:

- The definition of security policies for web architectures, especially on the client-side.
- Formally-proven analyses of web programming languages.
- Multi-level enforcement mechanisms for the security policies (based on static and dynamic analysis encompassing application-level and system-level software).

ASCOLA members are mainly interested in JavaScript as a programming language as well as the use of aspects as a seamless path from the definition of security policies and their composition to their implementation.

This year, we have finalized our proposal of extensible JavaScript modules and applied it to extend in a modular way the full-blown JavaScript interpreter Narcissus with several dynamic analyses including information-flow analyses.

9.2.2. ANR

9.2.2.1. SONGS (ANR/INFRA)

Participants: Adrien Lebre [coordinator], Jonathan Pastor, Anthony Simonet.

The SONGS project (Simulation of Next Generation Systems) is an ANR/INFRA project running for 48 months (starting in January 2012 with an allocated budget of 1.8MEuro, 95KEuro for ASCOLA).

The consortium is composed of 11 academic partners from Nancy (AlGorille, coordinator), Grenoble (MESCAL), Villeurbanne (IN2P3 Computing Center, GRAAL/Avalon - LIP), Bordeaux (CEPAGE, HiePACS, RUNTIME), Strasbourg (ICPS - LSIIT), Nantes (ASCOLA), Nice (MASCOTTE, MODALIS).

The goal of the SONGS project (http://infra-songs.gforge.inria.fr) is to extend the applicability of the SimGrid simulation framework from Grids and Peer-to-Peer systems to Clouds and High Performance Computation systems.

9.2.3. FSN

9.2.3.1. OpenCloudware (FSN)

Participants: Jean-Marc Menaud [coordinator], Thomas Ledoux.

The OpenCloudware project is coordinated by France Telecom, funded by the French Fonds National pour la Société Numérique (FSN, call Cloud n°1) and endorsed by competitiveness clusters Minalogic, Systematic and SCS. OpenCloudware is developed by a consortium of 18 partners bringing together industry and academic leaders, innovative technology start-ups and open source community expertise. The project started in 2012 for a duration of 42 months.

The OpenCloudware project aims at building an open software engineering platform, for the collaborative development of distributed applications to be deployed on multiple Cloud infrastructures. It will be available through a self-service portal. We target virtualized multi-tier applications such as JavaEE - OSGi. The results of OpenCloudware will contain a set of software components to manage the lifecycle of such applications, from modelling(Think), developing and building images (Build), to a multi-IaaS compliant PaaS platform (Run).

The ASCOLA project-team is mainly involved in the sub-projects "Think" (SLA model across Cloud layers) and "Run" (virtual machine manager for datacenters and placement constraints). The team has developed btrCloudStack, a private cloud based on the OpenSource CloudStack and integrating the work on placement rules and energy optimization. This software system has been extended this year.

9.2.3.2. Hosanna (FSN)

Participants: Jean-Marc Menaud [coordinator], Rémy Pottier.

The Hosanna project (aims to scientifically and technically addresses the problem of deploying applications on a distributed multi-cloud virtual infrastructure (private cloud, Amazon, OVH, CloudWatt, Numergy etc.). This recent need is an important topic issue highlighted by recent major Outages in 2013 by the biggest players in the cloud such as Amazon or Netflix. This project aims to provide services that allow users to deploy their cloud multi-tier applications on hybrid Clouds infrastructures without any separation between IaaS. The Ascola team is extending its optimization solution to address the task placement problem in a multi-cloud environment and will develop a case study on a secure distributed file system. The project started in 2015 for a duration of 2 years.

9.2.4. CPER

9.2.4.1. SeDuCe

Participants: Jean-Marc Menaud [coordinator], Adrien Lebre.

The SeDuCe project (Sustainable Data Centers: Bring Sun, Wind and Cloud Back Together), aims to design an experimental infrastructure dedicated to the study of data centers with low energy footprint. This innovative data center will be the first experimental data center in the world for studying the energy impact of cloud computing and the contribution of renewable energy (solar panels, wind turbines) as well on the scientific, technological, that economical. This project is integrated in the national context of grid computing (Grid'5000), and the Constellation project, which will be an inter-node (Pays de la Loire, Brittany). He also participated in the validation of scientific work in interdisciplinary axis STIC and energy efficiency of the laboratory of excellence COMIN Labs.

9.2.5. Inria Project Labs

9.2.5.1. DISCOVERY

Participants: Ronan Alexandre Rcherreau, Adrien Lebre [coordinator], Anthony Simonet, Mario Südholt.

To accommodate the ever-increasing demand for Utility Computing (UC) resources, while taking into account both energy and economical issues, the current trend consists in building larger and larger Data Centers in a few strategic locations. Although such an approach enables UC providers to cope with the actual demand while continuing to operate UC resources through centralized software system, it is far from delivering sustainable and efficient UC infrastructures for future needs.

The DISCOVERY initiative [36] aims at exploring a new way of operating Utility Computing (UC) resources by leveraging any facilities available through the Internet in order to deliver widely distributed platforms that can better match the geographical dispersal of users as well as the ever increasing demand. Critical to the emergence of such locality-based UC (also referred as Fog/Edge Computing) platforms is the availability of appropriate operating mechanisms. The main objective of DISCOVERY is to design, implement, demonstrate and promote a new kind of Cloud Operting System (OS) that will enable the management of such a large-scale and widely distributed infrastructure in an unified and friendly manner.

The consortium is composed of experts in the following research areas: large-scale infrastructure management systems, networking and P2P algorithms. Moreover, two key network operators, namely Orange and RENATER, are involved in the project.

By deploying and using a Fog/Edge OS on backbones, our ultimate vision is to enable large parts of the Internet to be hosted and operated by its internal structure itself: a scalable set of resources delivered by any computing facilities forming the Internet, starting from the larger hubs operated by ISPs, governments and academic institutions, to any idle resources that may be provided by end users.

ASCOLA leads the DISCOVERY IPL and contributes mainly around two axes: VM life cycle management and security concerns.

9.2.6. InriaHub

9.2.6.1. MERCURY

Participants: Ronan-Alexandre Rcherrueau, Adrien Lebre [coordinator].

ASCOLA, in particular within the framework of the DISCOVERY initiative has been working on the massively distributed use case since 2013. With the development of several proof-of-concepts around OpenStack, the team has had the opportunity to start an InriaHub action. Named MERCURY, the goal of this action is twofold: (i) support the research development made within the context of DISCOVERY and (ii) favor the transfer toward the OpenStack community.
Further information available at: http://beyondtheClouds.github.io.

9.3. European Initiatives

9.3.1. FP7 & H2020 Projects

9.3.1.1. CoqHoTT

Title: Coq for Homotopy Type Theory

Programm: H2020

Type: ERC

Duration: June 2015 - May 2020

Coordinator: Inria

Inria contact: Nicolas TABAREAU

Every year, software bugs cost hundreds of millions of euros to companies and administrations. Hence, software quality is a prevalent notion and interactive theorem provers based on type theory have shown their efficiency to prove correctness of important pieces of software like the C compiler of the CompCert project. One main interest of such theorem provers is the ability to extract directly the code from the proof. Unfortunately, their democratization suffers from a major drawback, the mismatch between equality in mathematics and in type theory. Thus, significant Coq developments have only been done by virtuosos playing with advanced concepts of computer science and mathematics. Recently, an extension of type theory with homotopical concepts such as univalence is gaining traction because it allows for the first time to marry together expected principles of equality. But the univalence principle has been treated so far as a new axiom which breaks one fundamental property of mechanized proofs: the ability to compute with programs that make use of this axiom. The main goal of the CoqHoTT project is to provide a new generation of proof assistants with a computational version of univalence and use them as a base to implement effective logical model transformation so that the power of the internal logic of the proof assistant needed to prove the correctness of a program can be decided and changed at compile time—according to a trade-off between efficiency and logical expressivity. Our approach is based on a radically new compilation phase technique into a core type theory to modularize the difficulty of finding a decidable type checking algorithm for homotopy type theory. The impact of the CoqHoTT project will be very strong. Even if Coq is already a success, this project will promote it as a major proof assistant, for both computer scientists and mathematicians. CoqHoTT will become an essential tool for program certification and formalization of mathematics.

9.3.1.2. BigStorage

Title: BigStorage: Storage-based Convergence between HPC and Cloud to handle Big Data

Programm: H2020

Duration: January 2015 - December 2018

Coordinator: Universidad politecnica de Madrid

Partners:

Storage Research Group, Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Ca Technologies Development Spain (Spain)

Commissariat A L Energie Atomique et Aux Energies Alternatives (France)

Deutsches Klimarechenzentrum (Germany)

ICS, Foundation for Research and Technology Hellas (Greece)

Fujitsu Technology Solutions (Germany)

Johannes Gutenberg Universitaet Mainz (Germany)

Universidad Politecnica de Madrid (Spain)

Seagate Systems Uk (United Kingdom)

Inria contact: G. Antoniu & A. Lebre

The consortium of this European Training Network (ETN) 'BigStorage: Storage-based Convergence between HPC and Cloud to handle Big Data' will train future data scientists in order to enable them and us to apply holistic and interdisciplinary approaches for taking advantage of a data-overwhelmed world, which requires HPC and Cloud infrastructures with a redefinition of storage architectures underpinning them - focusing on meeting highly ambitious performance and energy usage objectives. There has been an explosion of digital data, which is changing our knowledge about the world. This huge data collection, which cannot be managed by current data management systems, is known as Big Data. Techniques to address it are gradually combining with what has been traditionally known as High Performance Computing. Therefore, this ETN will focus on the convergence of Big Data, HPC, and Cloud data storage, ist management and analysis. To gain value from Big Data it must be addressed from many different angles: (i) applications, which can exploit this data, (ii) middleware, operating in the cloud and HPC environments, and (iii) infrastructure, which provides the Storage, and Computing capable of handling it. Big Data can only be effectively exploited if techniques and algorithms are available, which help to understand its content, so that it can be processed by decision-making models. This is the main goal of Data Science. We claim that this ETN project will be the ideal means to educate new researchers on the different facets of Data Science (across storage hardware and software architectures, large-scale distributed systems, data management services, data analysis, machine learning, decision making). Such a multifaceted expertise is mandatory to enable researchers to propose appropriate answers to applications requirements, while leveraging advanced data storage solutions unifying cloud and HPC storage facilities.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Organisation

10.1.1.1. Member of the Steering and Organizing Committees

- A. Lebre took part to the organisation of the Grid'5000 school in Grenoble (70 attendees).
- A. Lebre took part to the organisation of the workshop "Stockage informatique" during the Journées Scientifiques event in Nantes.
- J. Noyé was a co-organizer of LaMod '16, a workshop on Language Modularity co-located with Modularity '16 in Málaga, Spain.

10.1.2. Scientific Events Selection

10.1.2.1. Member of the Conference Program Committees

- J.-C. Royer was a member of the program committes WETCIE 2016, CAL 2016, ICIS 2016, IWAISE 2016, IIAI 2016.
- T. Ledoux was member of the program committees of the following workshops: Greens'16@ICSE, ARM'16@Middleware, CrossCloud'16@EuroSys.
- A. Lebre was a member of the program committees of ACM/IEEE CCGRID 2016, Europar 2016, ACM/IEEE SC 2016, IEEE CloudCom 2016, IEEE SSS 2016, OPTIM 2016.
- J. Noyé was a member of the program committee of Modularity '16 (Málaga, Spain).

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

- A. Lebre is associate editor for the IEEE Transactions on Big Data journal.
- M. Südholt is joint editor-in-chief of the journal Transactions on Modularity and Software Composition (Springer), formerly Transactions on AOSD.
- M. Südholt is an associate editor of the Journal on Programming, an open access journal.

10.1.3.2. Reviewer - Reviewing Activities

- A. Lebre has been a reviewer for the IEEE TPDS and IEEE TCC Journals, the IEEE Cloud Computing magazine, and the Journal of Parallel and Distributed Computing.
- T. Ledoux has been a reviewer for the IEEE Communications Letters.
- J. Noyé has been a reviewer for the Journal of Object Technology and Science of Computer Programming.

10.1.4. Invited Talks

- A. Lebre and Anthony Simonet have been invited to the 9th edition of the CloudControl Workshop (Sweden).
- A. Lebre has been invited to the BigStorage Initial Training Schoom (Spain).

10.1.5. Leadership within the Scientific Community

- A. Lebre is leading the OpenStack "Massively Distributed Working Group" (further information at: https://wiki.openstack.org/wiki/Massively_Distributed_Clouds).
- A. Lebre is member of the executive committee of the GDR CNRS RSD (Reseau et Système distribué). He is also co-leading the transversal action Virtualization and Clouds of this GDR.
- A. Lebre is member of the executive and architect committees of the Grid'5000 GIS (Groupement d'intérêt scientifique).
- T. Ledoux is member of the board of the Green Lab Center association. This association promotes and disseminates Green IT practices and research prototypes to the world of education, research and companies [1].
- M. Südholt is a member of the steering committees of the two international conferences on Programming and Modularity.
- M. Südholt has been a member of the board of the Aspect-Oriented Software Association.

10.1.6. Research Administration

- Pierre Cointe has been the head of the Lina laboratory that managed research in Computer Science of the main research institutions in Nantes.
- Jacques Noyé is the deputy head of the Mines Nantes department of Automation, Production and Computer Sciences.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

The team is involved in the following undergraduate and graduate-level programs at Mines Nantes and University of Nantes (the institutions all of eaching staff belongs to):

- The team is a main contributor to the **engineering program of EMN**.
- Within this engineering program, the team is steering, chairing and the main contributor to a two-year **graduate-level informatics specialization**. H. Grall is managing this program.
- The team is leading a three-year **engineering program on software engineering**. T. Ledoux is managing this program.

The team has also been involved in the following MSc programs that have been carried out with partners from French and foreign universities:

- The team participates in the **MSc program "Alma"** on software architecture and distributed systems, a joint program steered by colleagues from University of Nantes. In this context, we are responsible for a 48-hour module on advanced software composition and take part in the program's governing board. M. Südholt is managing the participation of Mines Nantes in this program.
- J.-C. Royer was teaching "Architecture, component programming and OSGi", from March 7 until 12, level M1, at the University of Monastir (Tunisia).

[1] Green Lab Center

m members have taught for about 220 hours on average in 2015 (hours of presence in front of students). Hereby, we have taken into account that researchers and some professors have not taught at times. In addition, another significant part of the program is taught by temporary staff, whose participation is managed by ASCOLA members.

10.2.2. Juries

- J.-C. Royer was reviewer of the HDR of Mohamed Bhiri (Université Grenoble), September 7, 2016. He was also member of the PhD defense of Jonathan Pépin (Université de Nantes) December 5, Madhi Benmoussa (Université Paris XIII) December 6, and Amine Benelallam (Ecole des Mines) December 7.
- A. Lebre was a member of the PhD committee of Vincent Kherbache, “Ordonnancement des migrations a` chaud de machines virtuelles”, Université de Nice - Sophia Antipolis, Dec. 7.
- T. Ledoux was a member of the PhD committee of Zakarea Al Shara (Univ. Montpellier), Nov. 17.

10.3. Popularization

- A. Lebre has been invited to the CargoDays event (Nantes).

11. Bibliography

Major publications by the team in recent years

[1] B. De Fraine, E. Ernst, M. Südholt. *Essential AOP: The A Calculus*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", December 2012, http://hal.inria.fr/hal-00676082

[2] I. Figueroa, T. Schrijvers, N. Tabareau, É. Tanter. *Compositional Reasoning About Aspect Interference*, in "Modularity'14: 13th International Conference on Modularity", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00919935

[3] M. S. Hasan, Y. Kouki, T. Ledoux, J.-L. Pazat. *Exploiting Renewable sources: when Green SLA becomes a possible reality in Cloud computing*, in "IEEE Transactions on Cloud Computing", July 2015, vol. PP, n^{o} 99, 1 p. [*DOI :* 10.1109/TCC.2015.2459710], https://hal.archives-ouvertes.fr/hal-01187907

[4] G. Jaber, G. Lewertowski, P.-M. Pédrot, M. Sozeau, N. Tabareau. *The Definitional Side of the Forcing*, in "Logics in Computer Science", New York, United States, May 2016 [*DOI :* 10.1145/2933575.2935320], https://hal.archives-ouvertes.fr/hal-01319066

[5] Y. Kouki, F. Alvares De Oliveira Jr., S. Dupont, T. Ledoux. *A Language Support for Cloud Elasticity Management*, in "CCGrid'14: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing", Chicago, United States, May 2014, pp. 1-8, https://hal.archives-ouvertes.fr/hal-00941945

[6] A. Lebre, J. Pastor, M. Südholt. *VMPlaceS: A Generic Tool to Investigate and Compare VM Placement Algorithms*, in "Europar 2015", Vienne, Austria, August 2015, https://hal.inria.fr/hal-01159033

[7] R. Pottier, J.-M. Menaud. *TrustyDrive: a Multi-Cloud Storage Service that Protects Your Privacy*, in "IEEE 9th International Conference on Cloud Computing", San Francisco, United States, International Conference on Cloud Computing, June 2016, https://hal.inria.fr/hal-01322638

[8] M. SOZEAU, N. TABAREAU. *Universe Polymorphism in Coq*, in "ITP'14: Interactive Theorem Proving", Vienna, Austria, July 2014, https://hal.inria.fr/hal-00974721

[9] N. TABAREAU, M. SÜDHOLT, É. TANTER. *Aspectual Session Types*, in "Modularity'14 - 13th International Conference on Modularity", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00872791

[10] J. M. VAN HAM, G. SALVANESCHI, M. MEZINI, J. NOYÉ. *JEScala: Modular Coordination with Declarative Events and Joins*, in "Modularity'14 - 13th International Conference on Modularity", Lugano, Switzerland, E. ERNST (editor), April 2014, https://hal.inria.fr/hal-00862332

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] R.-A. CHERRUEAU. *A Compositional Language of Security Techniques for Information Privacy in the Cloud*, Ecole des Mines de Nantes, November 2016, https://tel.archives-ouvertes.fr/tel-01416166

[12] S. DUPONT. *Crosslayer elasticity management for Cloud : towards an efficient usage of Cloud resources and services*, Ecole des Mines de Nantes, April 2016, https://tel.archives-ouvertes.fr/tel-01344377

[13] F. MARCHAND DE KERCHOVE DE DENTERGHEM. *Extending interpreters by diverting, or how to extend interpreters without modifying their source code*, Ecole des Mines de Nantes, November 2016, https://tel.archives-ouvertes.fr/tel-01415588

[14] J. PASTOR. *Contributions to massively distributed Cloud Computing infrastructures*, Ecole des Mines de Nantes, October 2016, https://tel.archives-ouvertes.fr/tel-01416099

[15] N. TABAREAU. *Managing Logical and Computational Complexity using Program Transformations*, université de nantes, November 2016, Habilitation à diriger des recherches, https://tel.archives-ouvertes.fr/tel-01406351

Articles in International Peer-Reviewed Journals

[16] N. BELDICEANU, M. CARLSSON, R. DOUENCE, H. SIMONIS. *Using finite transducers for describing and synthesising structural time-series constraints*, in "Constraints", January 2016, vol. 21, n° 1 [*DOI :* 10.1007/s10601-015-9200-3], https://hal.inria.fr/hal-01370322

[17] N. BELDICEANU, B. DUMAS FERIS, P. GRAVEY, S. HASAN, C. JARD, T. LEDOUX, Y. LI, D. LIME, G. MADI-WAMBA, J.-M. MENAUD, P. MOREL, M. MORVAN, M.-L. MOULINARD, A.-C. ORGERIE, J.-L. PAZAT, O. H. ROUX, A. SHARAIHA. *Towards energy-proportional Clouds partially powered by renewable energy*, in "Computing", January 2017, vol. 99, n° 1, 20 p. [*DOI :* 10.1007/s00607-016-0503-z], https://hal.inria.fr/hal-01340318

[18] I. FIGUEROA, N. TABAREAU, É. TANTER. *Effect capabilities for Haskell: Taming effect interference in monadic programming*, in "Science of Computer Programming", April 2016, vol. 119, pp. 3-30 [*DOI :* 10.1016/j.scico.2015.11.010], https://hal.inria.fr/hal-01400002

[19] D. SERRANO, S. BOUCHENAK, Y. KOUKI, F. ALVARES DE OLIVEIRA JR., T. LEDOUX, J. LEJEUNE, J. SOPENA, L. ARANTES, P. SENS. *SLA guarantees for cloud services*, in "Future Generation Computer Systems", January 2016, vol. 54, pp. 233–246 [*DOI :* 10.1016/j.future.2015.03.018], https://hal.archives-ouvertes.fr/hal-01162654

International Conferences with Proceedings

[20] E. ARAFAILOVA, N. BELDICEANU, R. DOUENCE, P. FLENER, M. A. FRANCISCO RODRÍGUEZ, J. PEARSON, H. SIMONIS. *Time-Series Constraints: Improvements and Application in CP and MIP Contexts*, in "CPAIOR 2016 - 13th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming", Banff, Canada, C.-G. QUIMPER (editor), Lecture Notes in Computer Science, Springer, May 2016, vol. 9676, pp. 18-34 [*DOI :* 10.1007/978-3-319-33954-2], https://hal.inria.fr/hal-01355262

[21] J. COHEN. *Renaming Global Variables in C Mechanically Proved Correct*, in "Fourth International Workshop on Verification and Program Transformation", Eindhoven, Netherlands, April 2016, https://hal.archives-ouvertes.fr/hal-01277269

[22] P.-E. DAGAND, N. TABAREAU, É. TANTER. *Partial Type Equivalences for Verified Dependent Interoperability*, in "ICFP 2016 - 21st ACM SIGPLAN International Conference on Functional Programming", Nara, Japan, September 2016, pp. 298-310 [*DOI :* 10.1145/2951913.2951933], https://hal.inria.fr/hal-01328012

[23] S. DUPONT, S. BOURI, F. ALVARES DE OLIVEIRA, T. LEDOUX. *ElaScript: a DSL for Coding Elasticity in Cloud Computing*, in "32nd ACM Symposium on Applied Computing - Track on Cloud Computing", Marrakesh, Morocco, Proceedings of the 32nd ACM Symposium on Applied Computing - Track on Cloud Computing, April 2017, https://hal.archives-ouvertes.fr/hal-01400236

[24] A. GARNIER, J.-M. MENAUD, N. MONTAVONT. *Bringing Complex Event Processing into Multitree Modelling of Sensors*, in "International Conference on Distributed Applications and Interoperable Systems (DAIS)", Heraklion, Greece, June 2016, https://hal.inria.fr/hal-01322670

[25] M. S. HASAN, F. ALVARES DE OLIVEIRA, T. LEDOUX, J.-L. PAZAT. *Enabling Green Energy awareness in Interactive Cloud Application*, in "IEEE International Conference on Cloud Computing Technology and Science 2016", Luxembourg, Luxembourg, December 2016, https://hal.inria.fr/hal-01365230

[26] G. JABER, G. LEWERTOWSKI, P.-M. PÉDROT, M. SOZEAU, N. TABAREAU. *The Definitional Side of the Forcing*, in "Logics in Computer Science", New York, United States, May 2016 [*DOI :* 10.1145/2933575.2935320], https://hal.archives-ouvertes.fr/hal-01319066

[27] A. LEBRE, J. PASTOR, A. SIMONET, F. DESPREZ. *Revising OpenStack to Operate Fog/Edge Computing infrastructures*, in "IEEE International Conference on Cloud Engineering", Vancouver, France, April 2017, https://hal.inria.fr/hal-01273427

[28] Y. LI, A.-C. ORGERIE, J.-M. MENAUD. *Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a Cloud data center*, in "PDP 2017 - 25th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing", St Petersburg, Russia, March 2017, https://hal.inria.fr/hal-01432752

[29] A. LÈBRE, A. SIMONET, A.-C. ORGERIE. *Deploying Distributed Cloud Infrastructures: Who and at What Cost?*, in "Proceedings of the fifth IEEE International Workshop on Cloud Computing Interclouds, Multiclouds, Federations, and Interoperability", Berlin, Germany, Intercloud 2016, April 2016, 6 p. [*DOI :* 10.1109/IC2EW.2016.48], https://hal.inria.fr/hal-01404594

[30] R. POTTIER, J.-M. MENAUD. *TrustyDrive: a Multi-Cloud Storage Service that Protects Your Privacy*, in "IEEE 9th International Conference on Cloud Computing", San Francisco, United States, International Conference on Cloud Computing, June 2016, https://hal.inria.fr/hal-01322638

[31] J.-C. ROYER, A. SANTANA DE OLIVEIRA. *AAL and Static Conflict Detection in Policy*, in "15th International Conference on Cryptology and Network Security", Milan, Italy, S. FORESTI, G. PERSIANO (editors), Cryptology and Network Security, Springer, November 2016, n° 10052, pp. 362-382 [*DOI :* 10.1007/978-3-319-48965-0_22], https://hal.archives-ouvertes.fr/hal-01396376

Conferences without Proceedings

[32] B. CONFAIS, A. LÈBRE, B. PARREIN. *Performance Analysis of Object Store Systems in a Fog/Edge Computing Infrastructures*, in "CloudCom", Luxembourg, Luxembourg, December 2016, https://hal.archives-ouvertes.fr/hal-01397686

[33] B. CONFAIS, A. LÈBRE, B. PARREIN. *Quel système de stockage pour les architectures Fog ?*, in "Compas'2016", Lorient, France, July 2016, https://hal.archives-ouvertes.fr/hal-01376292

Scientific Books (or Scientific Book chapters)

[34] Y. KOUKI, F. ALVARES DE OLIVEIRA, T. LEDOUX. *Cloud Capacity Planning and Management*, in "Encyclopedia of Cloud Computing", Wiley-IEEE Press, July 2016, https://hal.archives-ouvertes.fr/hal-01342153

Research Reports

[35] R.-A. CHERRUEAU, A. LEBRE, D. PERTIN, A. SIMONET, M. SIMONIN. *ENOS: a HolisticFramework forConducting ScientificEvaluations of OpenStack*, Inria Rennes Bretagne Atlantique ; Nantes, November 2016, n° RT-0485, https://hal.inria.fr/hal-01415522

[36] A. LÈBRE, J. PASTOR, F. DESPREZ. *A Ring to Rule Them All - Revising OpenStack Internals to Operate Massively Distributed Clouds: The Discovery Initiative - Where Do We Are ?*, Inria, February 2016, n° RT-0480, pp. 1-24, https://hal.inria.fr/hal-01320235

Other Publications

[37] B. AHRENS, R. MATTHES, A. MÖRTBERG. *From signatures to monads in UniMath*, December 2016, working paper or preprint, https://hal.inria.fr/hal-01410487

[38] W. BENGHABRIT, H. GRALL, J.-C. ROYER. *Monitoring accountability policies with AccMon framework*, June 2016, GDR-GPL, Poster, https://hal.inria.fr/hal-01332040

[39] F. MARCHAND DE KERCHOVE, J. NOYÉ, M. SÜDHOLT. *Extensible Modules for JavaScript*, April 2016, 3 p. , SAC '16 - 31st Annual ACM Symposium on Applied Computing, Poster [*DOI :* 10.1145/2851613.2851958], https://hal.inria.fr/hal-01407340

[40] P.-M. PÉDROT, N. TABAREAU. *An Effectful Way to Eliminate Addiction to Dependence*, January 2017, working paper or preprint, https://hal.inria.fr/hal-01441829

References in notes

[41] M. AKŞIT, S. CLARKE, T. ELRAD, R. E. FILMAN (editors). *Aspect-Oriented Software Development*, Addison-Wesley Professional, September 2004

[42] C. ALLAN, P. AVGUSTINOV, A. S. CHRISTENSEN, L. HENDREN, S. KUZINS, O. LHOTÁK, O. DE MOOR, D. SERENI, G. SITTAMPALAM, J. TIBBLE. *Adding trace matching with free variables to AspectJ*, in "ACM Conference on Object-Oriented Programming, Systems and Languages (OOPSLA)", R. P. GABRIEL (editor), ACM Press, 2005

[43] R. ALLEN, D. GARLAN. *A Formal Basis for Architectural Connection*, in "ACM Transactions on Software Engineering and Methodology", July 1997, vol. 6, n^{o} 3, pp. 213–49

[44] J. H. ANDREWS. *Process-Algebraic Foundations of Aspect-Oriented Programming*, in "Proceedings of the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns", Lecture Notes in Computer Science, 2001, vol. 2192, pp. 187–209

[45] M. ARMBRUST, A. FOX, R. GRIFFITH, A. D. JOSEPH, R. KATZ, A. KONWINSKI, G. LEE, D. PATTERSON, A. RABKIN, I. STOICA, M. ZAHARIA. *A view of cloud computing*, in "Communications of the ACM", 2010, vol. 53, n^{o} 4, pp. 50–58

[46] T. H. AUSTIN, C. FLANAGAN. *Multiple facets for dynamic information flow*, in "Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages", New York, USA, POPL '12, ACM, 2012, pp. 165–178, http://doi.acm.org/10.1145/2103656.2103677

[47] A. BELOGLAZOV, R. BUYYA. *Energy efficient resource management in virtualized cloud data centers*, in "in: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID'10, IEEE Computer Society", 2010, pp. 826–831

[48] L. D. BENAVIDES NAVARRO, M. SÜDHOLT, W. VANDERPERREN, B. DE FRAINE, D. SUVÉE. *Explicitly distributed AOP using AWED*, in "Aspect-Oriented Software Development (AOSD)", ACM Press, March 2006, pp. 51-62

[49] G. S. BLAIR, G. COULSON, P. ROBIN, M. PAPATHOMAS. *An architecture for next generation middleware*, in "Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing", Springer-Verlag, 1998

[50] A. BRACCIALIA, A. BROGI, C. CANAL. *A formal approach to component adaptation*, in "Journal of Systems and Software", 2005

[51] R. E. BROWN, E. R. MASANET, B. NORDMAN, W. F. TSCHUDI, A. SHEHABI, J. STANLEY, J. G. KOOMEY, D. A. SARTOR, P. T. CHAN. *Server and Data Center Energy Efficiency: Public Law 109-431*, in "Report to Congress", 06/2008 2008

[52] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI, T. REZK. *Session Types for Access and Information Flow Control*, in "CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings", P. GASTIN, F. LAROUSSINIE (editors), Lecture Notes in Computer Science, Springer, 2010, vol. 6269, pp. 237–252, http://dx.doi.org/10.1007/978-3-642-15375-4_17

[53] E. M. CLARKE, O. GRUMBERG, D. A. PELED. *Model Checking*, The MIT Press, Cambridge, Massachusetts, 1999

[54] A. COLYER, A. CLEMENT. *Large-scale AOSD for Middleware*, in "Proceedings of the 3rd ACM Int. Conf. on Aspect-Oriented Software Development (AOSD), Lancaster", K. LIEBERHERR (editor), ACM Press, 2004, pp. 56–65

[55] F. DEREMER, H. H. KRON. *Programming-in-the-large versus programming-in-the-small*, in "IEEE Transactions on Software Engineering", 1976, vol. SE-2, n^{o} 2, pp. 80-86

[56] G. DECKER, O. KOPP, F. LEYMANN, M. WESKE. *BPEL4Chor: Extending BPEL for Modeling Choreographies*, in "IEEE International Conference on Web Services (ICWS 2007)", IEEE Computer Society, 2007, pp. 296–303

[57] E. W. DIJKSTRA. *On the role of scientific thought*, in "Selected Writings on Computing: A Personal Perspective", Springer-Verlag, 1974, pp. 60–66, Published in 1982

[58] R. DOUENCE, P. FRADET, M. SÜDHOLT. *A framework for the detection and resolution of aspect interactions*, in "Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE'02)", Lecture Notes in Computer Science, Springer-Verlag, October 2002, vol. 2487, pp. 173–188, http://hal.inria.fr/inria-00072153

[59] R. DOUENCE, P. FRADET, M. SÜDHOLT. *Trace-Based Aspects*, in "Aspect-Oriented Software Development", M. AKŞIT, S. CLARKE, T. ELRAD, R. E. FILMAN (editors), Addison-Wesley, 2004, pp. 201-218

[60] R. DOUENCE, O. MOTELET, M. SÜDHOLT. *A formal definition of crosscuts*, in "Proceedings of the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns", Lecture Notes in Computer Science, Springer-Verlag, 2001, vol. 2192, pp. 170–186

[61] R. DOUENCE, D. LE BOTLAN, J. NOYÉ, M. SÜDHOLT. *Concurrent Aspects*, in "Proc. of the Int. ACM Conf. on Generative Programming and Component Engineering (GPCE)", ACM Press, October 2006, pp. 79-88

[62] H. FOSTER, S. UCHITEL, J. MAGEE, J. KRAMER. *Model-based Verification of Web Service Compositions*, in "Proceedings of the 18th IEEE Int. Conf. on Automated Software Engineering (ASE'03)", IEEE Computer Society, 2003, pp. 152–163

[63] I. FOSTER, Y. ZHAO, I. RAICU, S. LU. *Cloud computing and grid computing 360-degree compared*, in "Grid Computing Environments Workshop, 2008. GCE'08", Ieee, 2008, pp. 1–10

[64] A. FUGGETTA, G. P. PICCO, G. VIGNA. *Understanding Code Mobility*, in "IEEE Transactions on Software Engineering", May 1998, vol. 24, n^{o} 5, pp. 342–361

[65] GREENPEACE. *Make IT green: Cloud computing and its contribution to climate change*, Greenpeace International, March 2010

[66] K. HONDA, N. YOSHIDA, M. CARBONE. *Multiparty asynchronous session types*, in "Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco,

California, USA, January 7-12, 2008", G. C. NECULA, P. WADLER (editors), ACM, 2008, pp. 273–284, http://www.doc.ic.ac.uk/~yoshida/multiparty/multiparty.pdf

[67] G. KICZALES, E. HILSDALE, J. HUGUNIN, M. KERSTEN, J. PALM, W. G. GRISWOLD. *An Overview of AspectJ*, in "ECOOP 2001 — Object-Oriented Programming 15th European Conference, Budapest Hungary", Berlin, J. L. KNUDSEN (editor), Lecture Notes in Computer Science, Springer-Verlag, June 2001, vol. 2072, pp. 327–353, http://www.eclipse.org/aspectj/

[68] G. KICZALES. *Aspect Oriented Programming*, in "Proc. of the Int. Workshop on Composability Issues in Object-Orientation (CIOO'96) at ECOOP", July 1996, Selected paper published by dpunkt press, Heidelberg, Germany

[69] G. KICZALES, J. DES RIVIERES, DANIEL G. BOBROW. *The Art of the Meta-Object Protocol*, MIT Press, Cambridge (MA), USA, 1991

[70] J. KIENZLE, R. GUERRAOUI. *AOP - Does It Make Sense? The Case of Concurrency and Failures*, in "16th European Conference on Object-Oriented Programming (ECOOP'2002)", Malaga, Spain, B. MAGNUSSON (editor), Lecture Notes in Computer Science, Springer-Verlag, 2002

[71] T. LEDOUX. *OpenCorba: a Reflective Open Broker*, in "ACM Meta-Level Architectures and Reflection, Second International Conference, Reflection'99", Saint-Malo, France, P. COINTE (editor), Lecture Notes in Computer Science, Springer-Verlag, July 1999, vol. 1616, pp. 197–214

[72] M. MCILROY. *Mass produced software components*, in "Mass produced software components", Garmish, Germany, P. NAUR, B. RANDELL (editors), NATO Science Committee, October 1968, pp. 138-155

[73] N. MEDVIDOVIC, R. N. TAYLOR. *A Classification and Comparison Framework for Software Architecture Description Languages*, in "IEEE Transactions on Software Engineering", January 2000, vol. 26, n^{o} 1, pp. 70-93

[74] M. MERNIK, J. HEERING, A. M. SLOANE. *When and How to Develop Domain-Specific Languages*, in "ACM Computing Surveys", December 2005, vol. 37, n^{o} 4, pp. 316-344

[75] L. MIKHAJLOV, E. SEKERINSKI. *A study of the fragile base class*, in "A study of the fragile base class", Brussels, Belgium, E. JUL (editor), Lecture Notes in Computer Science, July 1998, vol. 1445, pp. 355-382

[76] D. H. NGUYEN, M. SÜDHOLT. *VPA-based aspects: better support for AOP over protocols*, in "4th IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)", IEEE Computer Society Press, September 2006

[77] O. NIERSTRASZ. *Regular Types for Active Objects*, in "Object-Oriented Software Composition", O. NIERSTRASZ, D. TSICHRITZIS (editors), Prentice Hall, 1995, chap. 4, pp. 99–121

[78] M. NISHIZAWA, S. CHIBA, M. TATSUBORI. *Remote Pointcut - A Language Construct for Distributed AOP*, in "Proceedings of the 3rd ACM Int. Conf. on Aspect-Oriented Software Development (AOSD), Lancaster", ACM Press, 2004

[79] D. L. PARNAS. *On the criteria for decomposing systems into modules*, in "Communications of the ACM", December 1972, vol. 15, n^{o} 12, pp. 1053-1058

[80] S. PEARSON. *Toward Accountability in the Cloud*, in "Internet Computing, IEEE", July-Aug. 2011, vol. 15, n^{o} 4, pp. 64-69, http://dx.doi.org/10.1109/MIC.2011.98

[81] F. PLASIL, S. VISNOVSKY. *Behavior Protocols for Software Components*, in "Transactions on Software Engineering", January 2002, vol. 28, n^{o} 9

[82] F. PUNTIGAM. *Coordination Requirements Expressed in Types for Active Objects*, in "ECOOP'97—Object-Oriented Programming", M. AKŞIT, S. MATSUOKA (editors), Lecture Notes in Computer Science, Springer-Verlag, 1997, vol. 1241, pp. 367–388

[83] N. SHARMA, S. BARKER, D. IRWIN, P. SHENOY. *Blink: managing server clusters on intermittent power*, in "SIGARCH Comput. Archit. News", March 2011, vol. 39, pp. 185–198, http://dx.doi.org/10.1145/1961295.1950389

[84] M. SHAW, D. GARLAN. *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996

[85] B. C. SMITH. *Reflection and Semantics in LISP*, Xerox Palto Alto Research Center, Palo Alto, 1984, n^{o} P84-00030

[86] S. SOARES, E. LAUREANO, P. BORBA. *Implementing distribution and persistence aspects with AspectJ* , in "Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications (OOPSLA-02)", C. NORRIS, J. J. B. FENWICK (editors), ACM SIGPLAN Notices, ACM Press, November 4–8 2002, vol. 37, 11, pp. 174–190

[87] S. SUNDARESWARAN. *Ensuring Distributed Accountability for Data Sharing in the Cloud*, in "Dependable and Secure Computing", 2012, vol. 9, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6165313

[88] R. J. WALKER, K. VIGGERS. *Implementing Protocols via Declarative Event Patterns*, in "Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE-12)", ACM Press, 2004, pp. 159 - 169

[89] M. WAND, G. KICZALES, C. DUTCHYN. *A Semantics for Advice and Dynamic Join Points in Aspect-Oriented Programming*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2004, vol. 26, n^{o} 5, pp. 890–910

[90] D. M. YELLIN, R. E. STROM. *Protocol specifications and component adaptors*, in "ACM Transactions of Programming Languages and Systems", March 1997, vol. 19, n^{o} 2, pp. 292–333

[91] Q. ZHANG, L. CHENG, R. BOUTABA. *Cloud computing: state-of-the-art and research challenges*, in "Journal of internet services and applications", 2010, vol. 1, n^{o} 1, pp. 7–18

[92] A. VAN DEURSEN, P. KLINT, J. VISSER. *Domain-Specific Languages: An Annotated Bibliography*, in "ACM SIGPLAN Notices", June 2000, vol. 35, n^{o} 6, pp. 26-36