Decentralized Self-Adaptation in Large-Scale Distributed Systems

Luca Florio[*] [†]
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milano, Italy
luca.florio@polimi.it

ABSTRACT

The evolution of technology is leading to a world where computational systems are made of a huge number of components spread over a logical network: these components operate in a highly dynamic and unpredictable environment, joining or leaving the system and creating connections between them at runtime. This scenario poses new challenges to software engineers that have to design and implement such complex systems. We want to address this problem, designing and developing an infrastructure, GRU, that uses self-adaptive decentralized techniques to manage large-scale distributed systems. GRU will help developers to focus on the functional part of their application instead of the needed self-adaptive infrastructure. We aim to evaluate our project with concrete case studies, providing evidence on the validity of our approach, and with the feedback provided by developers that will test our system. We believe this approach can contribute to fill the gap between the theoretical study of self-adaptive systems and their application in a production context.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General; C.2.4 [**Distributed Systems**]: Distributed Applications

Keywords

Decentralized, Self-Adaptive, Microservices, Docker

1. THE PROBLEM AND ITS IMPORTANCE

The problem we want to address is the effective application of self-adaptation to large scale distributed systems. The importance of this problem stems from the current IT scenario: the evolution of technology is leading to a world where each object around us is provided with some sort of "intelligence", making it able to compute data and act according to them, as well as to share these data with other objects. The increasing number of "intelligent" devices poses new challenges to software engineers that have to design and implement systems composed of a great number of elements interacting between them in different ways. Moreover, these systems work in a dynamic and unpredictable environment, and need the capability to adapt in an autonomous way to every change in the scenario in which they operate: they should be self-adaptive, meaning that they should be able to self-configure, self-heal and self-defend [16]. Thus, distributed self-adaptive systems are an actual need and require to be studied in order to understand the most effective way to design and manage such complex systems. However, while this aspect is addressed in theory, stable platforms and comprehensive software engineering approaches for these kinds of systems are still to come and their application in a "production context" is still missing. This research addresses these challenges aiming to simplify the application of decentralized self-adaptation to the design and development of distributed software systems that can benefit from it. We want to fill the gap between theory and practice, providing to practitioners a flexible and efficient solution they can adopt to quickly create an application that relies on a distributed infrastructure. In order to achieve this objective, we aim to develop an infrastructure to self-manage distributed systems that can be integrated seamlessly with existing ones, like Docker [1]. The application of self-adaptation to large-scale distributed systems is a challenging task. In particular some important aspects related to distributed systems need to be addressed:

- **Absence of a global knowledge.** The high number of nodes and their distribution make difficult to share the global status of the system.
- **Changing environments.** Large-scale distributed systems often present a high level of dynamism that cannot be controlled: components may join or leave the system at runtime changing their interconnections.
- **System stability and limited resource consumption.** The absence of global knowledge and the high dynamism can led the system to an unstable state, actuating unnecessary optimization actions and consuming precious computational resources.

[*]Advisor: Prof. Elisabetta Di Nitto
Email: elisabetta.dinitto@polimi.it

[†]The research reported in this article is partially supported by the European Commission grant no. FP7-ICT-2011-8-318484 (MODAClouds).

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
© 2015 ACM. 978-1-4503-3675-8/15/08...
http://dx.doi.org/10.1145/2786805.2803192

The application of centralized or hybrid strategies for self-adaptation in the described context can be difficult, due to the huge number of possible elements to manage and the highly dynamic context. Starting from this consideration, we focus our research on the study of a totally decentralized self-adaptation strategy. **We propose an infrastructure, GRU, that enables decentralized self-adaptation of any independently developed distributed software.** GRU supports self-adaptation of any application built as a composition of distributed and independent *microservices* executed within Docker containers. This approach involves both theoretical and practical challenges, such as the study of effective self-adaptation strategies based on partial knowledge and the implementation challenges related to distributed systems, which have been previously described.

2. RELATED WORK

Self-Adaptive systems have been widely studied in the past decade, starting from the Autonomic Computing manifesto by IBM [16]. The literature provides various approaches to deal with such systems. In particular, self-adaptation is addressed in various ways, ranging from centralized approaches [10, 11, 14], where one element is in charge and manage the others, to completely decentralised ones [6, 12, 17], where all elements are peers and the behaviour of the system emerges from the interaction of all the elements, through hierarchical approaches [4, 5, 15], where the elements are organised into a hierarchy. The great majority of studies on self-adaptive or autonomic systems have been evaluated through simulations or toy examples, without a complete empirical evaluation [18]. The main exception is the Rainbow framework [14], which has been used to convert a legacy system into a self-adaptive one in the work described in [9]. However, Rainbow addresses self-adaptation with a centralized approach, which may be not suitable for large scale distributed systems composed by a huge number of nodes. Self-adaptive techniques are used in Cloud Computing platforms to manage the scaling of computing instances: Google Cloud Platform takes care of the scaling of the system, which is completely transparent to the user; Amazon Web Services uses autoscaling policies defined by the user; Microsoft Azure allows the user to control the autoscaling system defining rules. These approaches, that usually are centralized or hierarchical, do not represent a complete autonomic or self-adaptive system.

3. CONTRIBUTIONS

The contributions of our work are both theoretical and practical, involving the study of decentralized self-adaptation and the creation of a software to apply this approach to a concrete application domain in a way that can be useful to practitioners.

3.1 Decentralized Self-Adaptation

The first contribution of our work is related to the application of a decentralized approach to self-adaptation for the management of a distributed system. In order to obtain decentralized self-adaptation, **we propose a solution based on the concept of self-organizing multi-agent systems** [19]. The system is managed by a set of autonomous agents that are able to interact between them and take local decisions from which the global behavior of the system emerges. The agents communicate through messages and can self-organize in logical groups of neighbors, exchanging information related to their internal state (following the "information sharing pattern" [20]). Self-adaptation is achieved by the implementation in each agent of an adaptation loop, that periodically runs and actuates the correct action accordingly to the internal state of the managed system and the partial information provided by neighbors. The adaptation loop embodies the MAPE-K loop (Monitor-Analyze-Plan-Execute on a Knowledge base) [16]. The agents act like a decentralized and distributed management layer and do not deal with the functional part of the system: agents act only as managers and mediators between the different services of the system, keeping it up and running and taking care of aspects like scalability, fault tolerance, performance, etc. The research question we want to answer with this contribution is the following:

- **Is a decentralized approach to self-adaptation like the one described in [19] suitable for a concrete implementation?**

3.2 Application Domain

The second contribution of our work regards the application of our decentralized self-adaptive approach to a concrete application domain. The proposed solution can be applied to large-scale distributed systems in general, but **we want to focus on the application of decentralized self-adaptation to the microservices architecture pattern [3] and Docker containers [1]**. This architectural pattern is used to build enterprise application deployed on cloud. Each microservice is considered as an autonomous and independent entity and can be implemented with any language, using the technology which is best suited for that kind of service. Services just need to expose an interface for communication. Microservices architecture is usually paired with Docker [1] containers: each microservice can be "dockerized" and run in a container as a process isolated from the others, having a restricted but direct access to the resources of the host (reducing the overhead of a traditional approach based on virtual machines). These technologies have been widely adopted in the last few years for building enterprise applications, even by big companies in the IT industry like Netflix [2]. The software infrastructure behind Netflix is totally based on microservices running in Docker containers deployed on the cloud. The use of this strategy allows Netflix to easily scale in order to satisfy the high number of requests from users as well as to manage possible failures in an effective way. Despite its advantages, the microservice architecture introduces additional complexity and new problems to deal with, most of them related to its distributed nature: the application is based now on a set of distributed elements which needs to communicate between them. Elements can join or leave the system dynamically, due to the scaling of the application or failures in a part of the network. These elements should be managed and deployed in an effective way. Self-adaptation could relieve the developers from the challenges that the implementation of this kind of systems implies, so we believe that microservices architecture based on Docker containers can be a good application domain for our project. The research question we want to answer with this contribution is the following:

- **Can we build a software to effectively apply decentralized self-adaptation to microservices running in Docker containers?**

4. EVALUATION OF THE PROJECT

The evaluation of the project is twofold: (i) we need to evaluate the capabilities of our solution to effectively manage a system composed of dockerized microservices; (ii) we need to evaluate if the proposed solution can be easily used by practitioners to help them build a distributed application based on Docker containers. **We plan to evaluate our approach based on decentralized self-adaptation addressing a concrete case study.** The target is the realization of a web application which provides some services to users. The application is composed by dockerized microservices and deployed on a cloud computing platform. The proposed system manages the evolution of the application, dealing with aspects like scaling, fault tolerance, performance, etc. in order satisfy the requests of the customers in a maximum response time defined in the Service Level Agreement. Using the described strategy we plan to evaluate:

- Ability of the system to scale according to the traffic load, without over- or under-provisioning of resources.
- Ability of the system to avoid waste of resources, allocating the correct amount of computational power for each service.
- Ability of the system to manage failures, keeping the application running with a negligible reduction in the quality of the service provided.
- Ability of the system to guarantee the quality of service defined in the Service Level Agreement .

The vast majority of self-adaptive systems have been evaluated through simulation or toy-examples [18]. However, we believe that an empirical evaluation based on a concrete case study like the one we described can represent a good evidence of the validity of our approach. **We also plan to evaluate how our solution can be integrated with tools and frameworks used by practitioners.** The creation of a self-adaptive system involves specific skills and knowledge, so the capability of running it as an external component that can be easily deployed to interact with the managed system is important to make self-adaptation applicable by practitioners. The evaluation will take place allowing a group of developers to test our system and collecting their feedback. This kind of evaluation can help to understand how self-adaptive solutions can be designed to best fit the needs of developers. in this way we address the following research question:

- **Can we provide a software that can be seamlessly integrated in a distributed system making it self-adaptive?**

5. CURRENT STATUS AND EVOLUTION

The first step in the realization of our project has been the study of decentralized self-adaptation applied to distributed systems. We started our research on the study of an existing prototype of self-adaptive decentralized systems based on the idea described in section 3.1, the SelfLet framework [13, 8, 7]. A SelfLet based system is composed by many SelfLets (i.e agents) spread over a logical

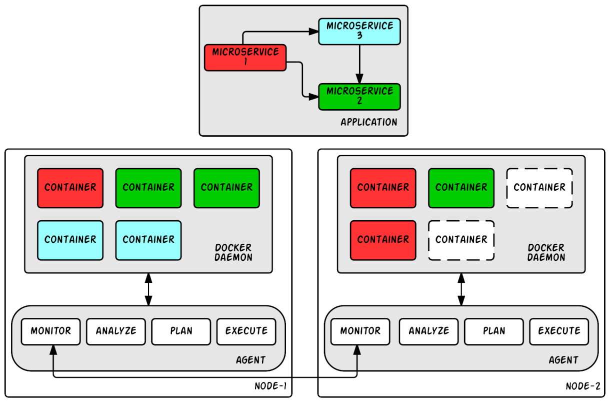

Figure 1: Architecture overview of GRU. The application is deployed using Docker containers: each container run a microservice, identified by a color.

network. Each SelfLet provides some services defined by the user and shares the same conceptual model and architecture. SelfLets communicate through messages dispatched by a message broker, self-organizing in groups that take decisions on the basis of the partial knowledge of the group itself. The behavior of the whole system emerges from the local decision taken by each group of SelfLets. Each SelfLet implements an adaptation loop, which allow the SelfLet to actuate an autonomic action on the basis of its internal state and the neighbors state. The SelfLet framework has been evaluated only through simulation. In order to verify if its theoretical basis could be applied to our research, we tested the SelfLet framework using a concrete case study. We built an application composed of more then fifty collaborating nodes and we deployed it on Amazon cloud infrastructure. We evaluated the capability of the system to handle a variable workload using self-adaptive strategies. We obtained promising results that prove the validity of our decentralized approach to self-adaptation and that we will describe in a future publication. The work on the SelfLet framework allowed us to better understand the issues related to this kind of systems, and to validate our decentralized approach to self-adaptation. We concluded that, despite the theoretical basis of the SelfLet framework is valid, it is not well suited to be used in a production context by practitioners.
Currently we are focusing on the application of decentralized self-adaptation to Docker containers. We are developing GRU[1], the infrastructure that will bring self-adaptation to Docker containers, making them able to self-manage. The target is the creation of a multi-agent system able to self-manage clusters of nodes on which are running multiple "dockerized" microservices (Fig. 1). Agents can interact with the Docker daemon, actuating autonomic actions (e.g. scaling, containers migration, etc.) on containers to keep the application up and running. Each agent implement the MAPE-K feedback loop, monitoring the status of the containers in its node and taking decisions on the actions to actuate starting from the information provided by the containers in the node and from other agents. Agents can self-organize in sets using a gossip protocol and communicate between them using RESTful API. GRU will be self-contained and

[1]https://github.com/elleFlorio/gru

able to seamlessly integrates with any application based on Docker containers, as well as with other tools that operates on containers (e.g. Swarm [1]). We are designing GRU on the basis of the SelfLet framework, with multiple agents spread over a logical network that communicate between them. However, while the SelfLet framework is based on an internal approach (i.e. the autonomic manager is part of the application itself), GRU provides an external approach, where each agent is an entity separated from the managed application. Despite this choice lead to several challenges related to the interaction with the managed system, using the external approach we pose less constraints on developers, which are free to build their application with the technologies and languages best suited for it. These are the steps we have identified to accomplish our goals:

1. Execution of a single agent on a single node.
2. Execution of multiple agents on multiple nodes.
3. Evaluation of the system on a concrete case study.
4. Test of the system by practitioners.
5. Evaluation of the feedback provided by practitioners.

The current status of the development is the execution of a single agent on a single node.

6. CONCLUSION

The goal of our research is to study how to apply decentralized self-adaptation to the development of large-scale distributed systems in a way that can contribute to fill the gap between the theoretical study of self-adaptation and its concrete adoption by practitioners. We want to focus on the emerging domain of microservices and Docker containers: our work can contribute to make easier the development of applications composed of "dockerized" microservices. To achieve our goal we are developing GRU, a system able to self-manage applications based on Docker containers that integrates seamlessly in the development of such applications.

7. REFERENCES

[1] Docker, (`https://www.docker.com/`).

[2] Microservices at netflix, (`http://www.slideshare.net/stonse/microservices-at-netflix`).

[3] Microservices, (`http://martinfowler.com/articles/microservices.html`).

[4] L. Baresi and S. Guinea. A3: self-adaptation capabilities through groups and coordination. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, pages 11–20, New York, NY, USA, 2011. ACM.

[5] S. Bouchenak, F. Boyer, B. Claudel, N. D. Palma, O. Gruber, and S. Sicard. From autonomic to self-self behaviors: The jade experience. *TAAS*, 6(4):28, 2011.

[6] T. Bures, I. Gerostathopoulos, P. Hnetynka, J. Keznikl, M. Kit, and F. Plasil. Deeco: An ensemble-based component system. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering*, CBSE '13, pages 81–90, New York, NY, USA, 2013. ACM.

[7] N. CALCAVECCHIA. *Decentralized management in distributed autonomic systems*. PhD thesis, Italy, 2013.

[8] N. Calcavecchia, D. Ardagna, and E. Di Nitto. The emergence of load balancing in distributed systems: the selflet approach. In D. Ardagna and L. Zhang, editors, *Run-time Models for Self-managing Systems and Applications*, Autonomic Systems, pages 97–124. Springer Basel, 2010.

[9] J. Cámara, P. Correia, R. De Lemos, D. Garlan, P. Gomes, B. Schmerl, and R. Ventura. Evolving an adaptive industrial software system to use architecture-based self-adaptation. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on*, pages 13–22. IEEE, 2013.

[10] S.-W. Cheng and D. Garlan. Stitch: A language for architecture-based self-adaptation. *Journal of Systems and Software*, 85(12):2860–2875, 2012.

[11] J. R. Cybrynski. Abc of the autonomic computing toolkit. Technical report, IBM Autonomic Computing Technical Report, 2005.

[12] F. De Pellegrini, D. Miorandi, D. Linner, L. Bacsardi, and C. Moiso. Bionets architecture: from networks to serworks. In *Bio-Inspired Models of Network, Information and Computing Systems, 2007. Bionetics 2007. 2nd*, pages 255–262, Dec. 2007.

[13] D. Devescovi, E. Di Nitto, D. Dubois, and R. Mirandola. Self-organization algorithms for autonomic systems in the selflet approach. In *Proceedings of the 1st International Conference on Autonomic Computing and Communication Systems*, Autonomics '07, pages 26:1–26:10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[14] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[15] E. Hoefig, B. Wuest, B. K. Benko, A. Mannella, M. Mamei, and E. Di Nitto. On concepts for autonomic communication elements. In *International Workshop on Modelling Autonomic Communications*, 2006.

[16] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[17] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. A multi-agent systems approach to autonomic computing. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 464–471. IEEE Computer Society, 2004.

[18] D. Weyns and T. Ahmad. Claims and evidence for architecture-based self-adaptation: a systematic literature review. In *Software Architecture*, pages 249–265. Springer, 2013.

[19] D. Weyns and M. Georgeff. Self-adaptation using multiagent systems. *Software, IEEE*, 27(1):86–91, 2010.

[20] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*, pages 76–107. Springer, 2013.