EPJ Web of Conferences **108**, 02029 (2016)
DOI: 10.1051/epjconf/201610802029
© Owned by the authors, published by EDP Sciences, 2016

Cloud Middleware Combining the Functionalities of Message Passing and Scaling Control

O. Iakushkin[1,a]

[1] *Department of Computer Modeling and Multi-Processing Systems, Saint Petersburg State University, Universitetskii prospect 35, 198504 Saint-Petersburg, Russia*

Abstract. The paper is focused on a solution providing middleware control over both message passing and scaling in a cloud environment. We provide a detailed analysis of the issue and present the restrictions particular to the load-balancing and communications systems. We also justify the need of such middleware development for a cloud infrastructure. The paper describes the problems emerged while working on a concrete project and presents the results obtained so far.

1 Introduction

While working on the "Computer technologies for monitoring and diagnostics of cultural heritage sites" and the "Research in the field of designing and implementing effective computational simulation for hydrophysical and hydro-meteorological processes of Baltic Sea (and the open Ocean and offshores of Russia)" projects, we needed a way to move the original system operating in a single cluster to a cloud computing environment including geo-distributed nodes. This paper describes some of the challenges we encountered in connection therewith.

The introduction of the 3rd Platform [1, 2] in many respects undermines the concept of powerful and autonomous P2P networks. The 3rd Platform virtually divides the computer world into 'weak' mobile clients and 'strong' servers. The distributed systems that provided non-interactive allocation of processing power are gradually replaced by cloud systems that allocate resources dynamically.

Contemporary cloud technologies enable to scale solutions within a data centre, but not only [3–6]. They also support geographical distribution of tasks and ensure information safety through duplication. Cloud computing resources offered by public providers encompass a wide range of specialized and standardized nodes with a variety of characteristics [7, 8], which is a distinctive feature of such providers. The heterogeneous computational resources are dynamically allocated within the system. Their use requires the development of interaction and management tools. In general, the geographical distribution and heterogeneity of the resources complicate the design of software solutions. At the same time, they provide flexibility and stable operation.

The transition to the 3rd Platform de facto means that most solutions should acquire the format of services. These are executed within computing resources that offer stability in conditions of load

[a] e-mail: oleg.jakushkin@gmail.com

increase. The so-called 'Fifth Wave of Innovation' is ending with a crisis: the superfluous flow of incoming data entails processing challenges and hinders the search of information crucial for particular tasks. The possible solutions are connected with the Big Data issue, which is a relatively new area.

The program complexes operating within the 2nd Platform need to be re-platformed for the service-oriented environment. Transition of each solution requires its tailoring to the new reality [9–11].

To that end, the major task is fragmented into separate components, or applications, that can interact in a cloud environment. This is how the service-based approach is implemented. The independent components of the executed task run in an autonomous way. They exchange data via a protocol which is known at least to the parties of information exchange. The operation can take place in one or more computing systems. The components can employ various platforms and interpretation environments. They might be written in different programming languages.

The interaction between the nodes of a distributed application is ensured by middleware [12–14]. The message exchange is not the only way to exercise control in cloud solutions [15, 16]. The arsenal also includes systems for dynamic launch/shut down of nodes and deployment of services. Message exchange would also be impossible without routing and addressing. The dynamic scaling of available resources poses the problem of interaction between the communications subsystem and the system used for components scaling. The solution of this problem allows to properly organise distributed and scalable heterogeneous systems that can accommodate load changes and employ components requiring no modification.

2 Description of the Examined Service System

Every node of a cloud system belongs to a certain 'class of equipment'. The nodes run applications (services), and the system can run multiple identical services. We presume the network connecting the nodes is reliable.

Services exchange messages of different types. Each message has a topic that serves as a marker. The messages are received by services with delay. The middleware acts as an external observer unaware of the logic which underlies the communication between the nodes. Each service can be characterized by the class of equipment it can run on and by the list of topics it is interested in.

There is also a pre-determined set of messaging patterns.

3 Bidirectional Patterns

1. The service requests data from services subscribed to a particular topic and gets a reply. The responding node is determined by a middleware-controlled algorithm (e.g., traditional Round-Robin that goes over all services subscribed to the topic, one by one).

2. The service can enter into a long-term dialogue with a particular service subscribed to the required topic. The latter node will continuously respond.

3. The service requests data from a number of services subscribed to a certain topic. It can set the time frame for response. The responding nodes will attempt to reply in time, but might fail to do so before the requesting service finishes its countdown.

In these patterns, requests are sent by means of messages. They are delivered within a certain period of time and characterized by both type and topic [17–20].

4 Unidirectional Patterns

1. The service can send messages to all services subscribed to a certain topic.
2. The service can send a message to one of the services subscribed to a certain topic.

5 Problems in Message Passing: The Middleware Perspective

The system allows the control over the nodes launching/shutting down and execution of service applications. Therefore, we need a strategy determining the choice of the receiving nodes. We should note that the nodes communicate according to the business logic that remains unknown at the level of the middleware. Likewise, the middleware cannot initiate nodes interaction. Moreover, there must be at least one service of each type at each point in time during the system operation.

At the same time, the middleware is responsible for the scaling strategy and selects where to deploy a new node and what parameters it will have. If the selection strategy does not conform with the message passing system, there arises a delay in the message delivery.

The system developers possess certain financial means that define the scope of the possible scaling at a particular point in time. The deployment of a service on the equipment of a particular class have a certain hourly cost. Besides, the pool of equipment is limited for each class.

The system nodes have a set of parameters that we know with certainty: the current load on the node (breakdown by CPU, GPU, RAM, and HDD); the number of messages sent and received for each topic and by each service; the number of requests the service has not yet responded; the number of each service's requests awaiting reply. This data is collected by nodes that have built-in middleware components. However, it is collected with some delay.

There is also another problem: some of the parameters can only be calculated with a limited degree of accuracy. These parameters include: the time of message passing between the nodes; the types of the messages; system indicators determined according to the logs analysis on a running system; for each message sent/received within the system, an array of parameters describing its content in the context of the running system.

All these problems set the task of developing a middleware combining scaling and message passing functionalities in a cloud infrastructure. Such middleware will allow to increase the volume of data transfer and minimize the operating cost.

6 Implemented Solution

We started the work by developing an architecture that allows direct message passing between the nodes. We introduced an outside controller that can expand the pool of available messaging patterns and monitor the work of the entire cloud system. Further details on the differences of our solution from the existent ones, and the main reasons behind why we started the project and its detailed structure are provided in [21–23]. Our current goal is to create a system enabling dynamic scaling of a distributed solution. The system will allow to perform a scaling based on the load within messaging patterns. We developed a novel protocol allowing us to meet the described goals of flexibility and extendibility (presented in figure 1) and created an open source project MQCloud[1] with its implementation.

[1] https://github.com/mqcloud/

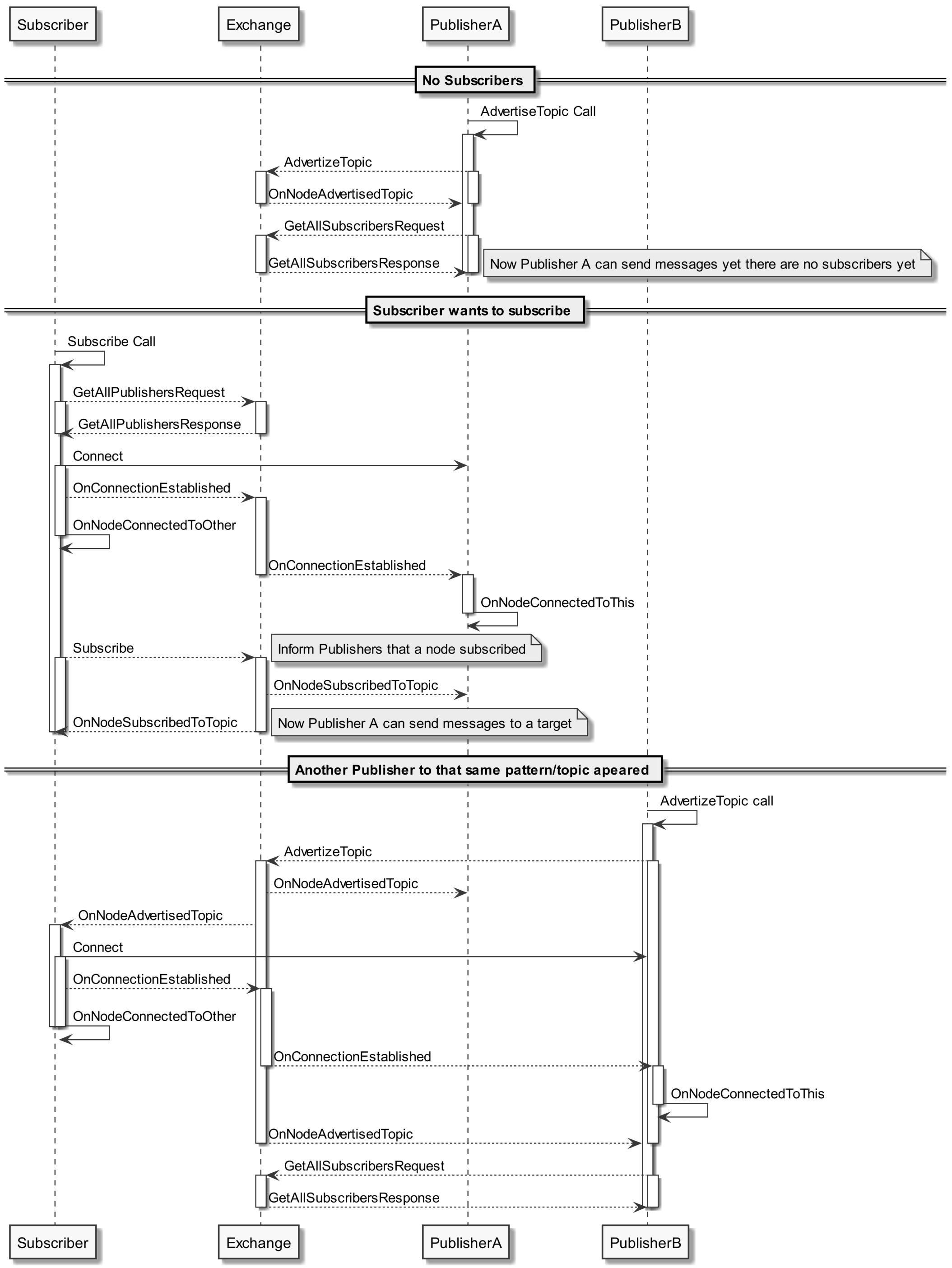

Figure 1. Nodes exchange protocol that can sustain common message passing patterns and be used to develop new once on top

Acknowledgements

The work has been done with partial support of SPbU (Saint Petersburg State University) grant 9.37.157.2014 and SPbU grant 0.37.155.2014. The research has been done using the computing resources of the resource center "SPbU Computing Facility".

References

[1] D. Oppermann, *A ICANN, o modelo multissetorial e o programa de novos dominios genericos (ICANN, the Multistakeholder Model and the New gTLD Program)* (Fonte, 2014), pp. 64–69

[2] M.A.d.S. Mendes, *Estilos arquiteturais web baseados em padroes abertos W3c* (Fonte, 2014), pp. 85–90

[3] I. Gankevich, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, V. Korkhov, A.B. Degtyarev, A.V. Bogdanov, V. Zolotarev, *Virtual private supercomputer: Design and evaluation*, in *Computer Science and Information Technologies (CSIT)* (IEEE, 2013), pp. 1–6, ISBN 978-1-4799-2460-8

[4] A.V. Bogdanov, A.B. Degtyarev, S.M. Lwin, T.K. Lwin, *Problems of Development of Complex Multilayered Applications in Distributed Environment*, in *Proceedings of the 4th Intern. Conf. "Distributed Computing and Grid-Technologies in Science and Education"* (JINR, Dubna, 2010), pp. 51–56, ISBN 978-5-9530-0269-1

[5] S.S. Ghag, R. Bandopadhyaya, in *Software Engineering Frameworks for the Cloud Computing Paradigm* (Springer, 2013), pp. 235–254, ISBN 978-1-4471-5030-5

[6] I. Gankevich, V. Korkhov, S. Balyan, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, A. Degtyarev, A. Bogdanov, in *Computational Science and Its Applications – ICCSA 2014*, edited by B. Murgante, S. Misra, A. Rocha, C. Torre, J. Rocha, M. Falcão, D. Taniar, B. Apduhan, O. Gervasi (Springer International Publishing, 2014), Vol. 8584 of *Lecture Notes in Computer Science*, pp. 341–354, ISBN 978-3-319-09152-5

[7] S. Doddavula, I. Agrawal, V. Saxena, in *Cloud Computing*, edited by Z. Mahmood (Springer, London, 2013), Computer Communications and Networks, pp. 197–219, ISBN 978-1-4471-5106-7

[8] E. Focht, A. Jeutter, in *Sustained Simulation Performance 2012* (Springer, 2013), pp. 51–64

[9] A. Dworak, F. Ehm, P. Charrue, W. Sliwinski, *The new CERN Controls Middleware*, in *Journal of Physics: Conference Series* (IOP Publishing, 2012), Vol. 396, p. 012017

[10] J. Lauener, *Rda3 transport*, http://indico.cern.ch/getFile.py/access?contribId=3&resId=1&materialId=slides&confId=259755 (2013)

[11] I. Yastrebov, *Rda3 high-level - api & architecture*, http://indico.cern.ch/getFile.py/access?contribId=3&resId=1&materialId=slides&confId=259755 (2013)

[12] W. Sliwinski, *Controls middleware renovation - technical overview*, http://indico.cern.ch/getFile.py/access?contribId=2&resId=1&materialId=slides&confId=259755 (2013)

[13] *Review of the controls middleware transport architecture and its use of zeromq*, http://indico.cern.ch/conferenceDisplay.py?confId=259755 (2013)

[14] A. Dworak, M. Sobczak, F. Ehm, W. Sliwinski, P. Charrue, *Middleware trends and market leaders 2011* (2011)

[15] D. Petcu, M. Rak, *Open-Source Cloudware Support for the Portability of Applications Using Cloud Infrastructure Services*, in *Cloud Computing* (Springer, 2013), pp. 323–341

[16] A.V. Bogdanov, A.B. Degtyarev, V. Mareev, N. Y., *Flexible Dynamic Pooling of Resources or Service-Oriented Grid Computing* (Institute of the Information Society, 2012), 2, pp. 61–70

[17] A. Videla, J.J. Williams, *RabbitMQ in action* (Manning, 2012)

[18] B. Snyder, D. Bosnanac, R. Davies, *ActiveMQ in action* (Manning, 2011)

[19] A.W. Group, *Amqp v1.0*, http://www.amqp.org/sites/amqp.org/files/amqp.pdf (2011)

[20] P. Hintjens, *ZeroMQ: Messaging for Many Applications* (O'Reilly, 2013), ISBN 978-1-4493-3404-8

[21] O. Iakushkin, V. Grishkin, *Unification of control in P2P communication middleware: Towards complex messaging patterns* (2015), Vol. 1648, pp. 1–4, http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.4912360

[22] O. Iakushkin, V. Grishkin, *Messaging middleware for cloud applications: Extending brokerless approach*, in *Emission Electronics (ICEE), 2014 2nd International Conference on* (2014), pp. 1–4

[23] V. Grishkin, O. Iakushkin, *Middleware transport architecture monitoring: Topology service*, in *Beam Dynamics and Optimization (BDO), 2014 20th International Workshop on* (2014), pp. 1–2