International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:12, No:2, 2018

A Design for Application of Mobile Agent Technology to MicroService Architecture

Masayuki Higashino, Toshiya Kawato, Takao Kawamura

***Abstract*—**A monolithic service is based on the N-tier architecture in many cases. In order to divide a monolithic service into microservices, it is necessary to redefine a model as a new microservice by extracting and merging existing models across layers. Refactoring a monolithic service into microservices requires advanced technical capabilities, and it is a difficult way. This paper proposes a design and concept to ease the migration of a monolithic service to microservices using the mobile agent technology. Our proposed approach, mobile agents-based design and concept, enables to ease dividing and merging services.

***Keywords*—**Mobile agent, microservice, web service, distributed system.

I. Introduction

With the evolution of cloud computing and web technology, many web services have been developed and operated which are updated frequently and continuously. Typically, many web services are built on a multi-layered architecture, and each layer has a monolithic architecture. However, the internal implementation of these layers is getting more complicated, and a wide area of rebuilds and redeployment may be required for updating the system. Operation of a system requiring frequent and continuous change requires a wide area of changes to be a burden of development and operation.

For this reason, it is necessary to localize the area of influence on the changed software module. Against this background, the usefulness of microservice architecture, which constructs a system by combining software components of microservices, is being evaluated [1].

There is currently no definition for microservices. According to [2], a single application is developed by combining multiple microservices. These microservices run in their processes and often communicate with each other by a lightweight communication protocol such as the REST API. Also, these microservices are built on the business capabilities and can be deployed independently by fully automated mechanisms. Centralized management for those services is minimal, and each service can use different programming languages and different data storage technologies. The features of such a microservice architecture are common to mobile agent technology, which are autonomous software components that can migrate among computers connected to the network, and are thought to have high affinity with each other.

Toshiya Kawato and Takao Kawamura are with Tottori University, Tottori, 680-8550, Japan.
Masayuki Higashino is with the Tottori University, Tottori, 680-8550, Japan (e-mail: higashino@tottori-u.ac.jp).

A mobile agent is an autonomous software module which can migrate between different computers via computer networks. A paradigm and a behavior of mobile agents are designed like humans and whose society such as collaboration and competition among people. This feature of mobile agents is assumed to contribute to the ease of management for microservices because microservices are hard to be managed these life cycle and relations among them as a distributed dynamic software module. The problem area of microservice architecture partly overlaps with mobile agent technology.

This paper discusses an design for application of mobile agent technology to microservice architecture and shows requirements and design for a mobile agent framework to manage microservices.

II. Requirements of Architecture

In general, web services are not designed with microservice architecture at the beginning of launching. Monolithic architectures and frameworks such as Ruby on Rails, CakePHP, etc. with high development efficiency are used early in many web services. It is difficult to predict in advance what feature is demanded at the time of launching the web service, making microservice at the time when necessity does not occur becomes a factor to raise the development cost of the system. Basically, disassembling software into many parts like a microservice is a trade-off between system complexity and maintainability, so it is common to adopt a monolithic architecture unless it is necessary. Thus, it can be said that one of the essential problems of microservice is a realization of easy change from monolithic architecture to microservice architecture.

In the following sections, we discuss requirements for simplifying changes from a monolithic architecture to microservice architecture using the paradigm of the mobile agent system.

A. Data Distribution

In the case of developing a web service, in recent years, relational databases, document databases, graph databases, key-value stores and the like are often adopted.

What is a problem when distributing these data to multiple computers (services) is to ensure data consistency. Generally, data stored in a logical or physically separate computer has a lower cost of ensuring consistency because latency and throughput of the network between computers are lower as the data is stored. Thus, the programmers of the system examine which data is strongly or weakly related to each other, and

adopts an approach in which the relationship between data is weak, or those with a low frequency of association are preferentially distributed. This approach is often adopted in the database architecture called NoSQL [3].

It is not unusual for the structure of data to change in the operation of the web service. In the microservice architecture, to localize the influence area of the maintenance of the system, an approach is taken to lower the degree of coupling between the services and increase the degree of condensation of the service. This is very similar to the optimization problem of the consistency assurance level strength and cost associated with the horizontal distribution of the database. In other words, it is thought that it is highly compatible with the architecture of microservices that localize the scope of influence of maintenance related to the service by putting it in a location that is closer to the relevant data.

Therefore, we propose to extract a data model with a strong association strength between models as a microservice while measuring the frequency of transactions of data generated by multiple data models and the communications traffic of data.

In addition, by enabling the operation to easily extract the microservice, it is thought that it becomes easy to change the monolithic web service to the microservice architecture.

A mobile agent is a useful aspect in cases where autonomy and transfer between computers are required. By providing a framework consisting of a suitable programming language and execution environment for this purpose, it is considered that the scalability of the system that can be easily realized can be achieved.

B. Process Distribution

To process distribution, those with less data input/output can be distributed relatively easily. On the other hand, a computer responsible for a large amount of data processing is strongly related to the data consistency guarantee mentioned above, so basically it is necessary to locate it close to the computer where the data is stored.

Therefore, the microservice having the function of processing data has the greatest influence on the network distance between the microservice having the data to be processed, the microservice as the processing result output object, and the end user's computers. It is required to migrate to a microservice with low cost.

Such a requirement is a field that the mobile agent is good at, and it seems that affinity between autonomous processing dispersion and mobile agent is high.

C. Domain-Driven Design

When dividing an existing service into microservice, a method of searching for a junction that can achieve both loose couplings and high condensability in the problem domain is used [1], [4].

However, this junction may go beyond the technical boundary. For example, in web services, a three-layer architecture is often adopted, but junctions that can achieve both loose coupling in the dispersion of data and dispersion of processing and high condensability traverse these layers there is a possibility.

Therefore, it is difficult to adopt the existing monolithic web application framework as it is. Many of these frameworks clearly divide user interface functions, data processing functions, and data storage functions into layers.

For this reason, by adopting a mobile agent, which is a small software component capable of both data retention and data processing, as a system platform, it is possible to realize a junction capable of achieving both loose couplings in the problem domain and high condensability It is thought that it can be flexible and easily extracted as a microservice.

Thus, we propose an approach that extracting microservices from a monolithic service and reconstructing a monolithic service to microservices dynamically.

III. Requirements of Framework

The smaller the service, the higher the cost for grasping the specification of each service and applying it accurately. For this reason, as compared to performing static specification verification in advance, the importance of being able to dynamically respond to changes in the system is increased when the operation of each microservice is changed.

Therefore, we propose a method to dynamically extract microservices from a monolithic service and dynamically reconstruct them as microservices. There are two important properties to realize this proposal.

A. Framework for Re-Construction

First, any part of a monolithic service or a microservice can be divided as a new microservice, and it works properly as a system. This can be realized by utilizing many existing virtualization technologies that have been proposed. However, the granularity of virtualization ranges from function of programming language to virtual machine. It is considered that mobility by mobile agent technology and correspondence to dynamic constraint satisfaction problem can be utilized for this.

Because relaxing restrictions on models based on business logic that developers think when dividing as microservices, and models based on hardware and technology being used, we can redefine from a more flexible monolithic service to microservice It is because our proposal intends to make the configuration possible.

For this reason, it is not an existing web framework based on restrictions of technical specifications such as a three-layer architecture and a request-response type such as an HTTP, rather than being as conscious as possible of restrictions on technical specifications, a more flexible monolithic service A new web framework with constraints and conventions that make it easier to split into microservices.

B. Evalution of Cost with Re-Comstruction

The second is to be able to evaluate the cost of extracting new microservice from a monolithic service. Even though it is possible to divide the system, the performance of the

system depends on constraints of the underlying hardware and software, so even if an advantage as microservices is obtained by division, estimation of performance degradation by division and measures to prevent irreversible division are extremely important.

Therefore, our proposal is based on the collection and analysis of statistical information on the inside of the service, based on the change of the performance cost when dividing microservices from a monolith service and the fact that it can be merged to the original system after being divided, In other words, it is necessary to think about a method to evaluate reversibility.

In our future work, we will proceed with these two studies.

IV. Design for Framework

A. Mobile Agent

A mobile agent architecture's specified in MASIF Specification by OMG (Object Management Group) [5] and Agent Management Specification by FIPA (Foundation for Intelligent Physical Agents) [6]. In AREs (agent runtime environment), multiple agents work concurrently and are able to migrate between AREs via networks. An agent is constructed from a runtime state area and an application area, and a program code area that contains program codes [7]. A runtime state area has information of states of the agent during executions of tasks such as call stacks, program counters, etc. An application area has any data that are specified by developers of the agent. A program code area has a set of program codes that are described behaviors of the agent.

B. Dividing and Mergeing for Processes

Fig. 1 shows a conceptual diagram of process dividing and merging. A programmer can migrate any process to any node. When a process is migrated, a local communication and a remote communication is handled transparently via the system. In general, a communication delay is greater in remote communication than in local communication. Thus, there is a tradeoff between redundancy due to system decentralization and communication delay by remote communication. In changing the construction of the system, it is important to balance the quality of the domain and the quality of the performance. In this our architecture, even if the process is migrated to an arbitrary node, interprocess communication is maintained whether it is local or remote. This makes it easier for programmers to find boundaries that can balance process dispersion and performance without being aware of communication types. This property is important in the domain-driven design, it makes easy to find for a better domain, and the changeability of the system can be improved.

C. Dividing and Mergeing for Data

Fig. 2 shows a conceptual diagram of resource dividing and marging. A programmer can clone arbitrary resources by duplicating or dividing it to an arbitrary number. Our framework is responsible for ensuring the connection with the

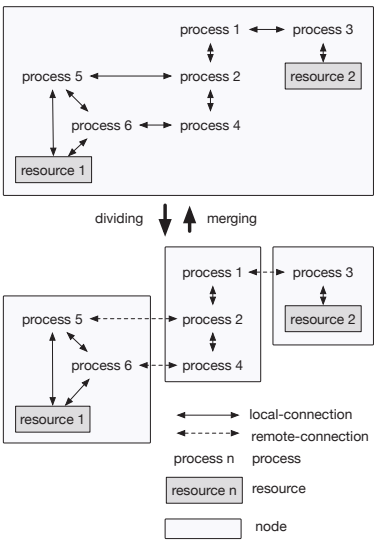

Fig. 1 A conceptual diagram of distributed dividing and merging of proesses

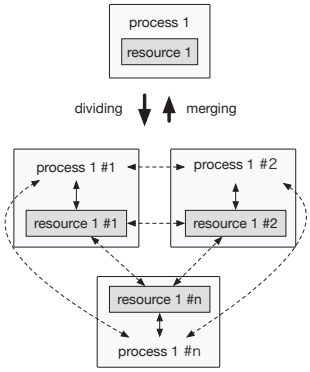

Fig. 2 A conceptual diagram of distributed dividing and merging of data (resource)

divided resources and an ability of look-up divided resources. Whether the divided resources take the master-slave model, multi-master model, Paxos [8] for consensus protocol or the like is realized by one layer above the framework. If a programmer just want to physically divide the resources, you can do so by dividing and merging of the process described above section.

In this our approach, by realizing the process distribution and the data distribution with different aspect, the programmer can understand separately the domain design of the program and the guarantee of consistency to the data.

V. Related Works

A. Mobile Agents and Microservices

Provalets [9] proposes a method to manage microservices using mobile agent technology. Fluid [10] proposes a transportable and adaptive web service model like a mobile agent architecture. Reference [11] proposes a Performance-based cost models for improving web service efficiency through dynamic relocation.

However, these are not has been made for processes to extract mictoservices from a monolithic service and reconstructing a monolithic service to microservices. Our proposed approach is extracting microservices from a monolithic service and reconstructing a monolithic service to microservices.

B. Web Services Composition and Microservices

Automatic web service composition like a mobile agent architecture, such as [12], is an active resaerch area [13]. Self-organization and self-management of processes including microservices too. In [14], associations of multiple microservices are explained by using a tree graph of services' types and graph trees of services' instances, and proposals are made to describe the concept of service orchestration and load balancing for automatic management of microservices that this work reuses much of the ideas from [15] described them. These approaches aim to composite existing services.

C. Cloud Computing

In the area of cloud computing, researchers are underway to adjust the physical location of virtual machines (VMs) for the quality of service (QoS) control. Reference [16] surveys QoS in cloud computing. Discussions related to mobile agents and microservices in the research area of cloud computing are focused on the migration of VMs such as [17]- [18]. In the live migration of a VM, migration is performed with the interfaces and various resources in VM being fragmented and active at the same time on a plurality of computers at the same time. This technology is probably considered to be useful as an idea for transparently dividing a monolithic service into multiple microservices and autonomous horizontal scaling. However, these live migration techniques for MVs do not assume that a VM will be divided into two or more instances during live migration.

D. Distributed Database

Reference [19] provides efficient partitioning and allocation of data for web service compositions. This approach that partitions and allocates small units of data, called micropartitions, to multiple database nodes, and improves data access efficiency over the standard partitioning of data. However, this approach is not premised on consistency with units of microservices based on business requirements which are important in microservice. However, it depends on the three-tier architecture, and the proposal of this paper, an extraction of microservices from a monolithic service, has not been proposed.

Azure Cosmos DB [20] is a globally distributed database that can choose a consistency level from strong, bounded staleness, session, consistent prefix, or eventual and choose a database model from collections, graphs, or tables.

On the other hand, our proposal is to easily divide monolithic web services into microservices while considering consistency with division units based on business requirements. In this case, in order to localize the scope of influence by service maintenance, it is the goal to dynamically lower the degree of coupling between services and to divide the service while increasing the degree of condensation of the service.

VI. Conclusions

This paper has discussed a design and concept using the mobile agent technology for deviding and merging between a monolithic service and microservices.

In future work, we design and develop the framework including programming language and its runtime platform for reconstructing microservices from a monolithic service.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 15K15982.

References

[1] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 2016.
[2] J. Lewis and M. Fowler. (2014) Microservices: a definition of this new architectural term. [Online]. Available: http://martinfowler.com/articles/microservices.html
[3] S. Edlich. (2017) Nosql databases. [Online]. Available: http://nosql-database.org/
[4] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003.
[5] *Mobile Agent System Interoperability Facilities Specification*, Object Management Group, Inc., 1997.
[6] *FIPA Agent Management Specification (SC00023K)*, Foundation for Intelligent Physical Agents, 2004.
[7] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," *IEEE Transactions on Software Engineering*, vol. 24, pp. 342–361, 1998.
[8] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.
[9] A. Paschke, "Provalets: Component-based mobile agents as microservices for rule-based data access, processing and analytics," *Business & Information Systems Engineering*, vol. 58, no. 5, pp. 329–340, 2016.
[10] I. M. D. Pratistha and A. Zaslavsky, "Fluid: Supporting a transportable and adaptive web service," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, 2004, pp. 1600–1606.
[11] D. Pratistha, A. Zaslavsky, S. Cuce, and M. Dick, "Performance based cost models for improving web service efficiency through dynamic relocation," in *Proceedings of the 6th International Conference on E-Commerce and Web Technologies*, 2005, pp. 248–257.
[12] P. Wang, Z. Ding, C. Jiang, M. Zhou, and Y. Zheng, "Automatic web service composition based on uncertainty execution effects," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 551–565, 2016.
[13] A. Immonen and D. Pakkala, "A survey of methods and approaches for reliable dynamic service compositions," *Service Oriented Computing and Applications*, vol. 8, no. 2, pp. 129–158, 2014.
[14] G. Toffetti, S. Brunner, M. Blöchlinger, F. Dudouet, and A. Edmonds, "An architecture for self-managing microservices," in *Proceedings of the 1st International Workshop on Automated Incident Management in Cloud*, 2015, pp. 19–24.
[15] G. Karagiannis, A. Jamakovic, A. Edmonds, C. Parada, T. Metsch, D. Pichon, M. Corici, S. Ruffino, A. Gomes, P. S. Crosta, and T. M. Bohnert, "Mobile cloud networking: Virtualisation of cellular networks," in *Proceedings of the 21st International Conference on Telecommunications*, 2014, pp. 410–415.
[16] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–17, 2014.
[17] B. Wei, C. Lin, and X. Kong, "Dependability modeling and analysis for the virtual data center of cloud computing," in *2011 IEEE International Conference on High Performance Computing and Communications*, 2011, pp. 784–789.

[18] M. Melo, P. Maciel, J. Araujo, R. Matos, and C. Arajo, "Availability study on cloud computing environments: Live migration as a rejuvenation mechanism," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2013, pp. 1–6.
[19] A. V. Kish, "Efficient partitioning and allocation of data for workflow compositions," Ph.D. dissertation, University of South Carolina, 2016.
[20] Microsoft Corporation. (2017) Azure cosmos db. [Online]. Available: https://azure.microsoft.com/en-us/services/cosmos-db/