International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 129-138
http://dx.doi.org/10.14257/ijdta.2014.7.4.10

Parallel Execution of Data-intensive Web Services Based on Data-flow Constructs and I/O Operation Ratio

Dongjin Yu[1], Qi Zhu[1], Jianhua Shao[2,3], Chang Li[1], Youwei Yuan[1] and Wanqing Li[1]

[1]*School of Computer, Hangzhou Dianzi University, Hangzhou, China*
[2]*Zhejiang Topcheer Information Technology Co., Ltd, Hangzhou, China*
[3]*Zhejiang Provincial Key Laboratory of Network Technology and Information Security*
yudj@hdu.edu.cn, zhuqi_409@163.com, sjh@topcheer.cn, cherry_728@163.com, yyw@hdu.edu.cn, liwanqing@hdu.edu.cn

Abstract

Data-intensive applications, such as data mining and data visualization, need to deal with huge amount of data. Service-oriented architecture, on the other hand, offers a scalable and flexible framework to implement loosely-coupled and standards-based distribute computing which the data-intensive applications usually require. In this paper, we apply four kinds of data-flow constructs, namely Map Construct, Reduce Construct, Conditional Construct and Loop Construct, to assist Web services composition to process huge volume of data. Furthermore, we put forward the approach to the composition of Web services based on the I/O operation ratio, in which the CPU intensive Web services and the I/O intensive ones are arranged to execute in parallel as far as possible. A case is presented finally to show that our approaches are feasible and effective.

Keywords: *Data-intensive, data-flow constructs, service composition, parallel, input and output operation ratio*

1. Introduction

The rapid growth of the Internet and World Wide Web led to vast amounts of information available online. In addition, business and government organizations create large amounts of both structured and unstructured information which needs to be processed, analyzed, and linked. The computing applications which require large volumes of data and devote most of their processing time to I/O and manipulation of data are deemed data-intensive [1]. On the other hand, Service-oriented architecture(SOA) is a widely accepted and engaged paradigm for the realization of business processes that incorporate several distributed, loosely coupled partners. It allows users to combine together fairly large chunks of functionality to form ad hoc applications built almost entirely from existing software services. Nowadays, the service-oriented approach using Web services and BPEL is of great interest for the implementation of data-intensive applications such as, for example, those found in the area of data analytics.

There is no doubt in the industry and research community that the importance of data intensive computing has been raising and will continue to be the foremost fields of research. As a result, the data intensive services based applications have become the most kind of application in SOA. Potentially, this could have a significant impact on the on-going researches for services and data intensive computing.

ISSN: 2005-4270 IJDTA

A lot of research achievements have been made in the field of data-intensive Web service. Many argue that the current technology of Web service is not suitable for data-intensive environment, and thus should be extended. Some propose the approaches to the data transmission among multiple data-intensive services [2-4]. Some introduce SOA-aware data-flow to optimize the management of data-incentive services in cloud [5-6]. There are also some researches on modeling, verification and discovery of data-intensive Web service [7-13]. However, all the researches mentioned above do not concern the parallel technology in service composition. Neither do they consider how the degree of parallelism affects the efficiency of the execution of the composite service.

In practical, the needs of users are met only by composing multiple elementary services. However, the technology of traditional relational data management fails to qualify itself for the processing of large-scale data among multiple services. Efficient parallel and concurrent technology is crucial in the analysis of large-scale data. Fortunately, the programming model of MapReduce with high fault tolerance and simple programming interface proposed by Google greatly simplifies the large-scale data processing in cluster. It means a lot by introducing the MapReduce programming model into the composite Web services to deal with large-scale data.

To tackle the problem brought by data-intensive Web Services, in this paper, we apply four kinds of data-flow constructs, namely Map Construct, Reduce Construct, Conditional Construct and Loop Construct, to assist Web services composition to deal with large-scale data. We also put forward the approach to the composition of Web services based on the I/O operation ratio, in which the CPU intensive elementary Web services and the I/O intensive ones are arranged to execute in parallel as far as possible.

The remaining of our paper is organized as follows. Section 2 presents the related work,whereas Section 3 introduces how to apply data-flow constructs to Web services and how to measure the throughput and cost of composite Web serviced based on data-flow constructs. After that, in section 4, we introduce the approach to the composition of Web services based on the I/O operation ratio. After Section 5 presents the case study, last section concludes the paper and outlines the future work.

2. Related Work

Service-oriented architectures and Web services are well-established paradigms for developing distributed applications. However, they face problems when accessing, moving and processing large datasets. Therefore, how to efficiently compose the Web services for data-intensive applications draws the most attention, especially on the approaches to the optimization and management of data migration among multiple Web services.

Deng *et al.*, propose a method for automatic data-intensive service composition, which can be executed in parallel [14]. They define the problem of automatic service composition by combing the approaches of State Space Search and Planning Graph. In addition, they propose a heuristic algorithm to compose data-intensive Web service automatically.

Zhou *et al.*, introduce an ontology-based approach to compose and publish data-intensive services for data aggregation [15]. They propose a rewriting-based query so as that the data service could dynamically adapt itself to clients' query request.

Modern applications of Web Services that involve the processing of large amounts of data tend to transmit data. In order to improve the performance of dynamic offloading strategy which might experience an overhead due to a prediction failure,

QuirinoZagarese *et al.*, quantitatively characterize the execution contexts that make dynamic offloading effective [4]. They also introduce the Attribute Loading Delegation technique that enables optimized data-transfers for those applications where data-intensive multiple-interactions take place. Meanwhile, Theodouli *et al.*, propose a specific approach to coupling performance control with congestion control features, in order to consider both performance and memory overflow issues in an integrated manner [16].

It is desirable to find the cost minimized service composition solution in service computing. Therefore, how to select appropriate data centers for accessing data replicas and how to select services with lowest associated costs are emerging problems when deploying and executing data-intensive service applications. In order to solve this problem, Wang et al propose a cost minimizing service composition model for data-intensive applications in [17].

As the data-intensive computing requires high throughput of IO, the data transfer from different node should be cut down as much as possible. In [18], Zhao *et al.*, propose a data-intensive parallel processing framework, which provides the flexible ability to support data distribution and allocation across the Internet, and semantics query.

Many architectures or real applications related with data-intensive applications have been proposed. Kowsar *et al.*, present the architecture of Cloud Man which supports not only the batch processing capability but also big data analysis workloads in the cloud environment [19]. There are also some real applications, such as Large-Scale Archive Access [20], Distributed Storage and Service for Spatial Data [18], *etc*.

Some researchers focus on combining Web service with MapReduce, which is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks [21]. For instance, Huang et al propose an automatic service composition method based on depth-first searching for the Top-k Qos service composition issue. They indicate that the method based on MapReduce can satisfy composition requirements quickly and efficiently even with a large-scale service repository [22]. Fei *et al.*, propose the concept of data-flow constructs based on MapReduce and some other data processing technology in [23] and [24]. In [25], Pan et al make use of skyline with MapReduce to select the appropriate services efficiently and equally. They show that their computing scheme of skyline with MapReduce outperforms traditional selection techniques in reducing the response time and sharing the computational workload more fairly. In [26], Hong et al present their Hadoop-based e-book Conversion System which transforms the image based e-books into EPUB-formatted e-books based on MapReduce. In this paper, however, we combine the data-flow constructs with services composition based on data-flow. Furthermore, our approach improves users' satisfaction of QoS.

3. Service Composition with Data-flow Constructs

Since data-intensive applications aim at the processing of large volume, it is advised that some data processing technology should be supportive to the dataflow-based Web service composition. Control-flow constructs such as If-Else are important, but far from efficient in data processing. In contrast to control-flow constructs, which are used to control and coordinate processes, data-flow constructs are designed for efficient and systematic data processing, including data parallelism and aggregation. We introduce the following 4 kinds of

data-flow constructs which are initially presented in the composition of scientific workflows by Fei in [23].

In this section, we illustrate how to calculate the response time and cost of the services with data-flow constructs applied. Fig. 1shows an example of composite service where the circle represents the elementary service, the box outside the elementary service represents applying a data-flow construct on the elementary service. The small box on the top right corner represents the exact construct applied, where C means *Conditional* construct, M, R and L means the *Map* construct, the *Reduce* construct and the *Loop* construct respectively. The dashed box groups the nodes which should execute in parallel, whereas the curving dashed line represents the loop execution of service based on the *Loop* construct. As shown in Figure 1, the services based on *Map* construct execute in parallel on multiple nodes that host multiple service replicas. Obviously, although parallel execution improves the efficiency and users thus get the optimal response time, the user have to pay much more for the service replicas.

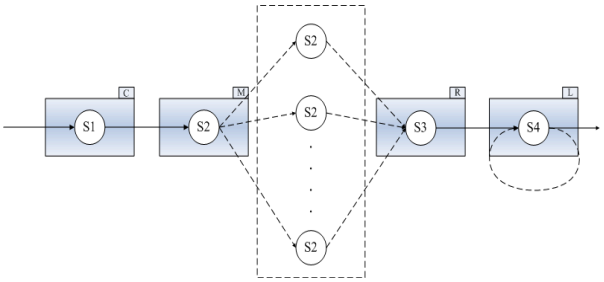

Figure 1. Composite Web Service Based on Data-flow Constructs

3.1. Map Construct

The *Map* construct enables the parallel processing of a list of data. Given an elementary service with multiple input ports and one output port, to apply the *Map* construct on one of input ports (map port)just means processing a list of data in parallel.

Suppose there exist multiple service replicas on n nodes after applying the *Map* construct on the elementary service S_i that is to be executed in parallel. We use *P(S_i, Map)*to represent the sum of the throughput of all nodes involved in the *Map* construct, and p_{ij} with $j = 1,2,\dots,n$to represent the throughput of the node of j. Thus,

$$P(S_i, Map) = \sum_{j=1}^{n} p_{ij} \qquad (1)$$

On the other hand, the response time of $S_i$with *Map* construct applied is given by:

$$T(S_i, Map) = \frac{1}{n}\sum_{j=1}^{n} t_{ij} \qquad (2)$$

wheret_{ij} means each node's response time in cluster.

Similar, the cost of S_i with *Map* construct is given by:

$$C(S_i, Map) = n * C_i \qquad (3)$$

To complete the parallel execution of the service S_i, the user must pay for all the parallel service replicas.

3.2. Reduce Construct

The *Reduce* construct enables the aggregation of a list of data items to a single data item. To apply the *Reduce* construct on an elementary service that has at least two input ports, an input port is designated as the reduce port, whereas another input port is designated as the base port. The reduce port takes input from the list of data items that need to be aggregated. On the other hand, the base port takes input either from an initial base data item or from the intermediate aggregation data item produced as the output of the previous iteration of aggregation.

The *Reduce* construct is used to aggregate data sequentially which implies the multiple service calls. Here, we assume the times of service invoking to be $r\ (r > 0)$. The response time of S_i with *Reduce* construct appliedis given by:

$$T(S_i, Reduce) = r * T_i \quad (4)$$

Similarly, the cost of S_i with *Reduce* construct can be calculated by:

$$C(S_i, Reduce) = C_i \quad (5)$$

3.3. Conditional Construct

The *Conditional* construct enables the conditional execution of a service based on a condition on one of the inputs. To apply the *Conditional* construct on an elementary service, one of the input ports is designated as the conditional port, on which a logical test will be evaluated based on the input data.

The *Conditional* construct enables the conditional execution of a Web service S_i. In other words,S_i can be executed only if the predicate p evaluates to be true.For simplicity, we do not take the time cost caused by the predicate evaluation into consideration. The response time of S_i with *Conditional* construct applied is given by:

$$T(S_i, Conditional) = T_i \quad (6)$$

Similarly, the cost of $S_i$with *Conditional* construct appliedis given by:

$$C(S_i, Conditional) = C_i \quad (7)$$

3.4. Loop Construct

The *Loop* construct enables the cyclic execution of an elementary service based on a predicate over the output of the service. To apply the *Loop* construct on a service, one of the input ports is designated as the loop port, which takes input either from the initial input data at the first iteration or the feedback from the output of the previous iteration. A predicate p specified by user is set on the output port o such that if p evaluates to be false, the output will be returned to the loop port.

The *Loop* construct is similar to the *Conditional* construct as they both have a predicate. However, the predicate in *Loop* construct is set on the output data from last iteration. The times of cyclic execution is therefore affected by both the predicate and the output data from last iteration. Here, we assume the iteration executes$l\ (l > 0)$times and the response time for each cyclic execution to be equal.

The response time of S_i with *Loop* construct is given by:

$$T(S_i, Loop) = l * T_i \quad (8)$$

Similarly, the cost of S_i with *Loop* construct is given by:

$$C(S_i, Loop) = C_i \quad (9)$$

4. Parallel Web Services Based on I/O Operation Ratio

The data-intensive Web services can be classified as the following two kinds: CPU intensive ones and I/O intensive ones. Here, CPU intensive Web services demand a lot of computation, whereas I/O intensive Web services do heavy read and write operations in the disk. Obviously, we can compose a composite Web service which contains both CPU intensive Web service and I/O intensive Web Service to achieve better performance.

The core of our approach lies in that the CPU intensive Web services and the I/O intensive ones should be arranged to execute in parallel as far as possible. In this way, it can avoid the competition of both CPU and I/O devices to the most extent. Table 1 illustrates the algorithm used to compose Web services in detail, in which we use I/O operation ratio to measure the extent of I/O intensiveness as described in (10).

$$r_{bws}^{cws} = \frac{f_{bws}}{P_{cws}} \times 100\% \quad (10)$$

wherer_{bws}^{cws} is the I/O operation ratio of the elementary Web service bws in the composite Web service cws, f_{bws} is the throughput of the elementary Web service bws and P_{cws} is the throughput of the composite Web service cws.

Table 1. The Algorithm Used to Compose Web Services Based on I/O Operation Ratio

1) Input the I/O operation ratio for all elementary Web services that form a composite Web service.
2) Mark all elementary Web services as $unmarked$ ones.
3) If there is no $unmarked$elementary Web service, go to Step 10.
4) Find the elementary Web service $ws_{io-lowest}$ that has the lowest I/O operation ratio among all $unmarked$elementary Web services.
5) Mark $ws_{io-lowest}$ as $marked$ one.
6) If there is no $unmarked$elementary Web service left, go to Step 10.
7) Find the elementary Web service $ws_{io-highest}$ that has the highest I/O operation ratio among all $unmarked$elementary Web services.
8) Mark $ws_{io-highest}$as $marked$ one.
9) If there are no data dependencies between $ws_{io-lowest}$and $ws_{io-highest}$, execute $ws_{io-lowest}$ and $ws_{io-highest}$ in parallel and go to Step 3. Otherwise mark $ws_{io-highest}$as $unmarked$ one and go to Step 4.
10) End.

5. Case Studies

We conducted the experiment on the Distributed Data Mining Platform developed by ourselves to validate our approach.The typical dataflow involves the tasks of data preprocessing, data classification and model presentation, which are encapsulated asdifferent Web services. Since the data classification involves huge amount of data, we apply the *Map* and *Reduce* constructs on the Web services of data classification. In addition, we apply *Conditional* constructs on those of data preprocessing and *Loop*

constructs on those of model presentation. We measured the runtime of the whole process by the varying number of nodes for data classification. The execution time in seconds for (i)the whole runtime, (ii)the preprocessing time, (iii)the classification time, and (iv)the time of the presentation of the result are presented in Figure 2.

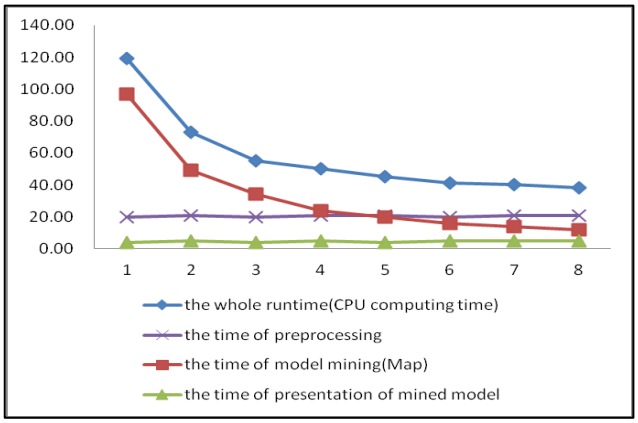

Figure 2. The Runtime Varied by the Different Number of Nodes

As shown in the Figure 2, each time we add one node, the total response time namely the CPU computing time is reduced. Since the data size is fixed, there's almost no change of time-consuming in the phase of the preprocessing and the presentation of the result.

5. Conclusion

Service-oriented architecture provides a scalable and flexible framework to implement loosely-coupled, standards-based, and protocol-independent distribute computing. The service-oriented approach using Web services is also of great interest for the implementation of data-intensive processes such as data mining, image processing and so on. In this paper, we combine the MapReduce programming model with Web services. Our work is the extended one introduced in [27] and [28]. More specifically, we apply the data-flow constructs including Map, Reduce, Conditional and Loop in the process of data-intensive service composition. Afterwards, we put forward the approach to the composition of Web services based on the I/O operation ratio, in which the CPU intensive elementary Web services and the I/O intensive ones are arranged to execute in parallel as far as possible.

In a nutshell, to apply the data-flow constructs, especially the *Map* and *Reduce* constructs, just means to distribute the tasks on multiple nodes, whereas to compose Web services based on I/O operation ratio just means to parallel the CPU-intensive and I/O intensive Web services in a single node. In the future, we will combine these two approaches and take our case as the reference in the optimization of more general cases. Moreover, we will conduct a more extensive experiment to demonstrate the effectiveness of our approaches.

Acknowledgments

The work is supported by Natural Science Foundation of Zhejiang (No.LY12F02003), the Key Science and Technology Project of Zhejiang (No. 2012C11026-3) and the open

project foundation of Zhejiang Provincial Key Laboratory of Network Technology and Information Security. The authors would also like to thank anonymous reviewers who made valuable suggestions to improve the quality of the paper.

References

[1] A.M. Middleton,"Data-Intensive Technologies for Cloud Computing", Handbook of Cloud Computing. Springer, **(2010)**.
[2] SpirosKoulouzis, R. Cushing, K. A. Karasavvas, A. Belloum, and M. Bubak,"Enabling Web Services to Consume and Produce Large Datasets", IEEE Internet Computing, vol.16, no.1, **(2012)**, pp.52-60.
[3] S. AmerYahia, Y. Kotidis,"A Web Services Architecture for Efficient XML Data Exchange",International Conference on Data Management,**(2004)**, pp.523-534.
[4] Q. Zagarese, G. Canfora, E. Zimeo and F. Baude,"Enabling Advanced Loading Strategies for Data Intensive Web Services",IEEE 19th International Conference on Web Services, **(2012)**, pp.480-487.
[5] D. Habich, W. Lehner, S. Richly and U. Assmann,"Using Cloud Technologies to Optimize Data-Intensive Service Applications",IEEE 3rd International Conference on Cloud Computing,**(2010)**, pp.19-26.
[6] Q. He, J. Han, Y. Yang, J. Grundy, "QoS-Driven Service Selection for Multi-tenant SaaS", IEEE 5th International Conference on Cloud Computing (CLOUD), **(2012)**, pp.566-573.
[7] J. Caverlee, L. Liu and D. Rocco,"Discovering and Ranking Data Intensive Web Services: A Source-Biased Approach", Georgia Institute of Technology CERCS technical report GIT-CERCS-03-26, **(2003)**.
[8] G.Wu, J. Wei, C. Ye, H. Zhong, Tao Huang and Hong He,"Specification and monitoring of data-centric temporal properties for service-based systems", The Journal of Systems and Software, vol.85,**(2012)**,pp.2738-2754.
[9] Y. Li and C. Lin,"QoS-Aware Service Composition for Workflow-Based Data-Intensive Applications",International Conference on Web Services,**(2011)**, pp.452-459.
[10] P. A. Bernstein, N. Dani, B. Khessib, R. Manne *et al.*, "Data Management Issues in Supporting Large-Scale Web Services", IEEE Data Eng. Bull., **(2006)**, pp.3-9.
[11] D.Habich, S. Richly and M.Grasselt,"Data-Grey-BoxWeb Services in Data-Centric Environments",International Conference on Web Services, **(2007)**, pp.976-983.
[12] X. Liu, Q. Zhao, G. Huang, H. Mei *et al*., "Composing Data-Driven Service Mashups with Tag-Based Semantic Annotations", International Conference on Web Services,**(2011)**, pp.243-250.
[13] C. Xu, W. Qu, H. Wang, Z. Wang *et al*., "A Petri Net-Based Method for Data Validation of Web Services Composition", COMPSAC, (2010), pp.468-476.
[14] S. Deng, L. Huang, B. Wu and L. Xiong,"Parallel Optimization for Data-Intensive Service Composition", Journal of Internet Technology, vol. 14, iss. 5, **(2013)**, pp.817-824.
[15] L. Zhou, H. Chen, Y. Mao,"Data service composition approach based on query rewriting", Computer Integrated Manufacturing Systems, vol.15, iss.4, **(2009)**, pp.823-832.
[16] A. Theodouli and A. Gounaris,"Adaptive memory-aware chunk sizing techniques for data-intensive queries over Web Services",28th Annual ACM Symposium on Applied Computing, **(2013)**, pp.826-831.
[17] L. Wang, J. Shen, C. Di, Y. Li *et al*.,"Towards minimizing cost for composite data-intensive services", IEEE 17th International Conference on Computer Supported Cooperative Work in Design, CSCWD,**(2013)**, pp.293-298.
[18] D. Zhao,Y. Gu and Z. Huang,"A new data-intensive parallel processing framework for spatial data", Lecture Notes in Electrical Engineering, vol.277, **(2014)**, pp.375-383.
[19] Y. Kowsar,E. Afgan,"Support for data-intensive computing with CloudMan", 36th International Convention on Information and Communication Technology, Electronics and Microelectronics, **(2013)**, pp.243-248.
[20] M. Tanaka, Y. Murakamiand K. Zettsu, "Data-Intensive Services for Large-Scale Archive Access", IEEE Ninth International Conference on Services Computing (SCC), **(2012)**,pp.617-624.
[21] J. Dean and S. Ghemawat. MapReduce,"Simplified data processing on large clusters",Communications of the ACM, vol.51 no.1, **(2008)**,pp.107-113.
[22] L. Huang, S. Deng, K. Dai,Y. Li *et al*.,"Automatic service composition in parallel with MapReduce",Acta Electronica Sinica, vol.40, iss.7, **(2012)**, pp.1397-1403.
[23] X. Fei, S. Lu and C. Lin,"A MapReduce-enabled scientific workflow composition framework", IEEE International Conference on Web Services,**(2009)**, pp.663-670.
[24] X. Fei and S. Lu,"A dataflow-based scientific workflow composition framework", IEEE Transactions on Services Computing, vol. 5, iss.1, **(2012)**, pp.45-58.
[25] L. Pan, L. Chen and J. Wu,"Skyline web service selection with MapReduce", International Conference on Computer Science and Service System (CSSS), **(2011)**, pp.739-743.

[26] T. HoHong, C. HoYun, J. WonPark, H. GeonLee *et al*., "Big data processing with MapReduce for e-book", International Journal of Multimedia and Ubiquitous Engineering, vol.8, iss.1, **(2013)**, pp.151-162.
[27] C. Li, D. Yu, Y. Yin, Y. Yuan et al., "Optimizing Web Service Composition in Parallel", Advanced Science and Technology Letters, vol.45,**(2014)**, pp.70-74.
[28] D. Yu, C. Li and Y. Yin., "Optimizing Web Service Composition for Data-intensive Applications",International Journal of Database Theory and Application, vol.7, iss.2, **(2014)**, pp.1-12.

Authors

Dongjin Yu, he is currently a professor at Hangzhou Dianzi University, China. He received his BS and MS in Computer Science and Technology from Zhejiang University in China, and PhD in Management from Zhejiang Gongshang University in China. His current research efforts include service computing, program comprehension and cloud computing. He is especially interested in the novel approaches to constructing large enterprise information systems effectively and efficiently by emerging advanced information technologies. The concern of his research closely relates with real applications of e-government and e-business. He is the director of Institute of Cloud Computing and Big Data, and the vice director of Institute of Intelligent and Software Technology of Hangzhou Dianzi University. He is also amember of ACM and IEEE, and a senior member of China Computer Federation (CCF).

Qi Zhu, he is currently is a postgraduate student in Hangzhou Dianzi University. His primary research areasincludesoftware archetecture and service computing.

Jianhua Shao, he was born in Jan. 1956, andreceived his bachelor's degree in Mathmatics from Fudan University in China. His primary research area includesnetworking computing and system integration.

Chang Li, she was born in July 1988, andcurrently is a postgraduate studentin Hangzhou Dianzi University. Her primary research area focuses on service computing and software engineering.

Youwei Yuan, he was born in 1966. He received his doctor degree in computer science from Wuhan University of Technology, China, in 2007. He is currently aprofessor of computer science, Hangzhou Dianzi University (Hangzhou, China). His research interests include artificial intelligence, data mining and distributed parallel processing, He has published over 50 technical papers in prestigious journals and conferences, 20 papers been indexed by SCI and EI.

Wanqing Li, he received the PhD degree in mechanics of solid from Lanzhou University in 2007 and works as an Associate Professor in School of Computer Science in Hangzhou Dianzi University. His present interests are numerical parallel computing and data mining.

