Alexandria Engineering Journal (2013) 52, 313–318

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

SODIM: Service Oriented Data Integration based on MapReduce

Ghada ElSheikh [a], Mustafa Y. ElNainay [a,*], Saleh ElShehaby [b], Mohamed S. Abougabal [a]

[a] *Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt*
[b] *Biomedical Engineering Department, Medical Research Institute, Alexandria University, Alexandria, Egypt*

Received 22 November 2012; revised 10 February 2013; accepted 28 February 2013
Available online 31 March 2013

KEYWORDS

Service Oriented Data Integration;
MapReduce;
Cloud Computing;
Web-services

Abstract Data integration has become a backbone for many essential and widely used services. These services depend on integrating data from multiple sources in a fast and efficient way to be able to provide the accepted level of service performance it is committed to. As the size of data available on different environments increases, and systems are heterogeneous and autonomous, data integration becomes a crucial part of most modern systems.

Data integration systems can benefit from innovative dynamic infrastructure solutions such as Clouds, with its more agility, lower cost, device independency, location independency, and scalability. This study consolidates the data integration system, Service Orientation, and distributed processing to develop a new data integration system called Service Oriented Data Integration based on MapReduce (SODIM) that improves the system performance, especially with large number of data sources, and that can efficiently be hosted on modern dynamic infrastructures as Clouds.

1. Introduction

Data integration has become a significant parameter in the information technology revolution. Therefore, data integration systems should serve the fast changing requirements and handle the complex computations and the large amounts of data in an adequate way. Subsequently, data integration systems must become more flexible, agile, reliable, and extensible. To achieve these requirements and face the challenges derived by the current dynamic environments, better utilization for the currently available techniques and technologies that can affect, support, and facilitate this goal is needed.

* Corresponding author. Tel.: +20 1003810376.
E-mail addresses: gh.elsheikh@gmail.com (G. ElSheikh), ymustafa@alexu.edu.eg (M.Y. ElNainay), sshehaby@mcit.gov.eg (S. ElShehaby), mohamed.abougabal@alexu.edu.eg (M.S. Abougabal).

Peer review under responsibility of Faculty of Engineering, Alexandria University.

Production and hosting by Elsevier

1110-0168 © 2013 Production and hosting by Elsevier B.V. on behalf of Faculty of Engineering, Alexandria University.
http://dx.doi.org/10.1016/j.aej.2013.02.007

Web-service is one of the emerged standards that shield the internal complexities of the integrated systems. Another approach known as Service Oriented Architecture (SOA) [1,2] became a significant force today in building large and loose-coupled integration systems with reusable components. On the hosting environments' level, new architecture has emerged providing computing resources and storage on demand and adopts the pay-as-you-go pattern in charging users known as Cloud Computing [3]. Clouds provide the illusion of endless resources and facilitate extending and shrinking resources as required by user. Clouds processing power which depends on a pool of machines orchestrated by Cloud providers encouraged the emergence of new processing frameworks such as MapReduce framework [4,5]. MapReduce framework is designed to handle large amounts of data on distributed environment, which makes it a good candidate to run over Clouds. The aim of the presented work is to combine web-service technology, SOA design paradigm, and MapReduce framework to develop a new data integration system called Service Oriented Data Integration based on MapReduce (SODIM). This should results in a system featuring more extendibility, agility, reliability, and fault tolerance.

The rest of this paper is organized as follows. Related work is surveyed in Section 2. The proposed system design description and architecture is presented in Section 3. Testing setup and results along with the related discussions are provided in Section 4. Finally, the conclusion is derived, and future extensions are suggested in Section 5.

2. Related work

Data integration focuses on allowing queries to be answered over a number of data sources. Many approaches were adopted in generating required mappings, all worked for the same goal but in different techniques. One adopted approach is Global-as-View (GAV) [6,7] that has a single mediated schema described as views (mappings) over local data sources. Another approach is Local-as-View (LAV) [8,9] which has the local data source being described as views over global schema. Different approaches were derived from LAV and GAV [10–12].

With the need for more flexible systems and with the evolution of SOA and Software as a Service concept SaaS [13], the efforts continued to produce more dynamic and flexible integration systems; such as Service Oriented Data Integration Architecture (SODIA) [14]. SODIA has been proposed to provide a dynamically unified view of data on demand from various autonomous, heterogeneous, and distributed data sources. Service providers publish their data sources as data access services, which may then be discovered, bound at the time they are needed and disengaged after use. Hence, the changes such as organization structures, backend data sources, data structures, and semantics could be managed dynamically and potentially reduce the maintenance cost. To achieve this, those data access services should be semantically described and discovered. SODIA system has limitations such as the delegation of mapping source data-elements to the integrated data-element to the service provider. Moreover, semantic descriptions for data access services (DASs) are left to service providers, therefore, there is no guarantee that the terms and conditions of the service could be interpreted correctly, and as a result, there is no guarantee that the requested service can be discovered and executed correctly.

On the other hand, with the explosion of data sizes and after the evolution of Cloud and Grid Computing, the Scale-out trend (add new workers) has been preferred over Scale-up (increase worker resources), and the idea of distributed processing gained larger interest. Different frameworks were used to process huge data sets on distributed sources such as Virtual Database system (VDB) using MapReduce framework [15]. The system utilizes the parallel and distributed processing capability of MapReduce framework to handle heterogeneous query execution on large datasets.

Each of the previously surveyed systems has its own desirable features. Thus, the need is emerged to provide a new data integration system called Service Oriented Data Integration based on MapReduce (SODIM) which enjoys the union of all desirable features of [14,15]. In addition, it should provide a unified form of data type transformation; therefore, it integrates data in more dynamic and agile manner. The features of existing and proposed systems are summarized in Table 1. The proposed system design is described in the next section.

Table 1 Proposed system versus existing systems surveyed in related work.

Properties	System		
	SODIA [14]	VDB [15]	SODIM
1. No global schema	√	X	√
2. No central data source model needed	√	X	√
3. Sources are semantically described and discovered	√	X	X
4. Unified data types transformations	X	X	√
5. Distributed processing supported	X	√	√
6. Multiple data source types support	√	√	√
7. Dynamic data source binding	√	X	√
8. System is loosely-coupled	√	X	√
9. Standardized integrated-data access (WSDL, SOAP)	√	X	√
10. Service oriented	√	X	√
11. Fault tolerance	X	√	√
12. Sources have unified interfaces (web-services)	√	X	√
13. Move code to data is enabled	X	√	√
14. Reliability management	X	√	√

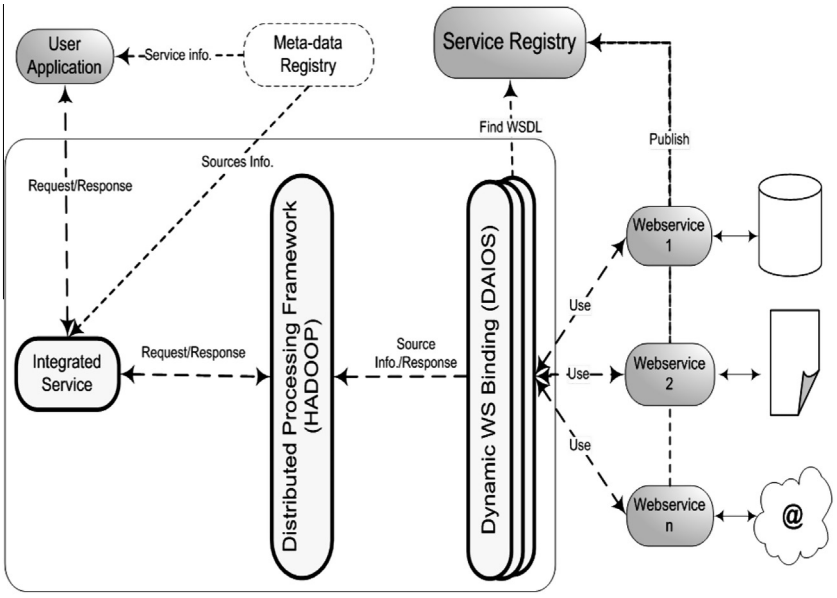


Figure 1 SODIM system architecture.

3. SODIM system design

SODIM system is designed as a pool of collaborative services spread on different interactive layers as shown in Fig. 1. These services are constructed according to SOA principles. It is also built on MapReduce distributed processing framework. Another part of the system is the late binding technique used to connect to data sources' services which insures more dynamism. The main parts of the system are represented in the following subsections.

3.1. Dynamic web-service binding

This component of the system architecture is responsible for executing dynamic binding to web-services, based on the service information provided on Web-Services Description Language (WSDL). It is designed on the notion of message exchange. It receives required information about a particular service from caller (such as WSDL location, Target Operation, Target Namespace) and uses this information to carry out service invocation without the need to pre-compiled service access components.

The DAIOS framework [16,17], which adopts messaging technique in communication, is picked for this task in the proposed system to achieve a unified form of data type transformation.

3.2. Distributed processing framework

The distributed processing framework is responsible for distributing the received job over worker nodes as smaller tasks and then aggregates results from related workers, combines them together, and then returns results back to caller. It carries out all complicated tasks of distributed processing and has full control on the participating (worker) nodes. It also provides a high level of reliability and fault tolerance on software level. The chosen framework implementation for this job is Apache Hadoop [4,5].

3.3. Metadata

Services metadata represents all needed data to allow accessing a specific service. Service-groups are the metadata carriers in the proposed system. Metadata registry can be implemented in different ways according to enterprise resources and policies (e.g., database, file service registry, etc).

3.4. Service registry

Service registry is the third party that allows service requester to discover service provider. The registry holds up-to-date information about businesses and the services they offer. This way, a service requestor can discover existing web-services, determines their purpose, functionality, and operation instructions and hence use a service to his benefit through its WSDL.

3.5. Integrated Service

The Integrated Service in Fig. 1 represents the business domain service provisioned to answer a specific business question. Integrated Service carries user question to the related sources and takes the responsibility to get and aggregate results in order to return them back to user.

The proposed system is expected to achieve all features presented in Table 1, and this will be due to combining the Service Oriented Data Integration and the distributed processing. Adopting dynamic binding of the Service Oriented Data Integration provides the loose coupling from source by adopting messaging techniques which facilitates adding or deleting a data source and makes the process of modifying integration system as a result of data source changes as easy as configuration modifications. This ensures minimal interruptions to running systems. Moreover, building Service Oriented Data Integration System on top of Hadoop framework shields the complexities of distributed processing and provides the "move code to data" capability that enables overcoming network-issues influence on system performance. It also provides more

reliability by data replication provided by Hadoop Distributed File System (HDFS) [5] on the software level rather than depending only on hardware solutions. The proposed system is implemented, and its performance is tested on real-life data. Testing setup environment and results are provided in the next section.

4. Results and discussion

The proposed system is implemented and tested on two different environment setups.

- *Single node test:* In this test, non-distributed SODIM (SODIA) is tested versus SODIM. The test is established on single node Hadoop setup.
- *Cloud test:* In this test, Amazon Elastic MapReduce (EMR) services have been used as a platform to test SODIM on cloud environment. Amazon EC2 has been used to provide the required infrastructure for different Hadoop cluster sizes as detailed later in this section.

Test results and discussions for each of the two test setups are provided in the following subsections.

4.1. Single node test results

In this test, both SODIA system and SODIM are implemented and tested on a single node. Test results for different numbers of data sources are shown in Fig. 2. According to the test results, some notices are listed below.

- SODIM gives longer execution time than SODIA system with small number of data sources. This is because of the overhead introduced by Hadoop platform processes, which becomes noticeable with small input data sizes.
- SODIM gives better execution time with large number of data sources, because of using multiple Maps, and executing tasks in parallel, even on the same Java Virtual Machine (JVM).

4.2. Cloud test results

By using Amazon EMR, different test types have been conducted with different clusters' sizes and different numbers of data sources. Multiple Job Flows [18–20] were created on Amazon EMR web-services. Some modifications to Hadoop configurations and tuning are applied as described below.

- In each cluster, input file is divided into $n - 1$ chunks (n = number of cluster nodes). Master node does not perform any processing tasks, and therefore, it is excluded from the calculations.
- In each cluster, data replication in HDFS is configured to be one (minimum).
- In each cluster, maximum number of MAP tasks allowed on each node is one to ensure that the job is evenly distributed over worker nodes.

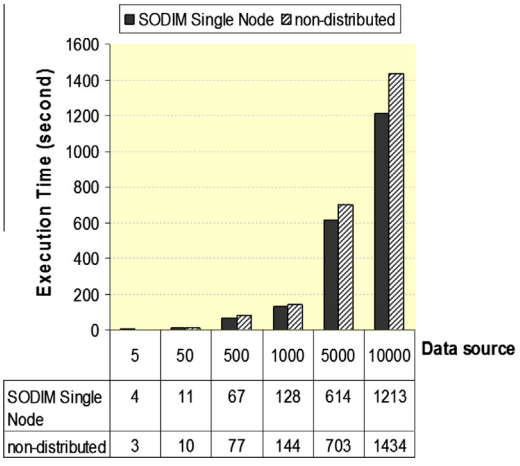
Applying the previously mentioned tuning parameters to Hadoop cluster gives the results shown in Fig. 3.

Three different types of tests were held on this environment.

- *Test A:* is a Hadoop cluster that consists of one master node and two slave nodes of type m1.small of Amazon instances.
- *Test B:* is a Hadoop cluster that consists of one master node and three slave nodes of type m1.small of Amazon instances.
- *Test C:* is a Hadoop cluster that consists of one master node and four slave nodes of type m1.small of Amazon instances.

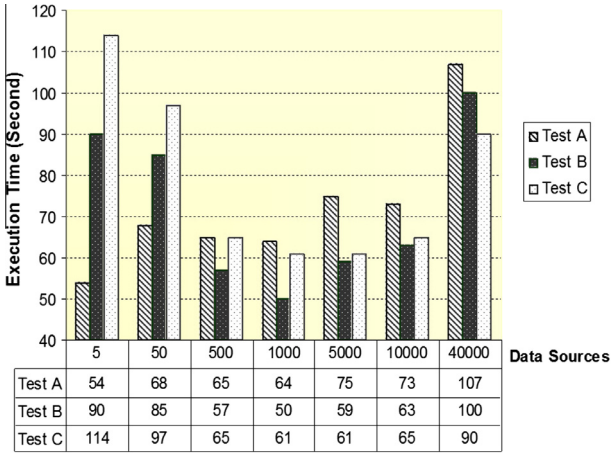
According to Tests' execution time, the following is noted for each test.

- *Test A:*
  - Test A gives its best execution time in the range (5–50) data sources compared to other tests, because it has the smallest number of cluster nodes, and therefore, the minimum cluster daemons overhead.



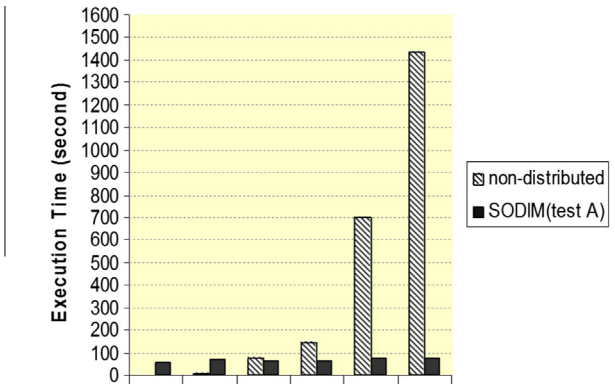
	5	50	500	1000	5000	10000
SODIM Single Node	4	11	67	128	614	1213
non-distributed	3	10	77	144	703	1434

Figure 2 SODIM versus non-distributed system (SODIA) on single node setup.



	5	50	500	1000	5000	10000	40000
Test A	54	68	65	64	75	73	107
Test B	90	85	57	50	59	63	100
Test C	114	97	65	61	61	65	90

Figure 3 SODIM test on different Hadoop cluster sizes on Amazon EMR after tuning.



	5	50	500	1000	5000	10000
non-distributed	3	10	77	144	703	1434
SODIM(test A)	54	68	65	64	75	73

Figure 4 SODIM (Test A) versus non-distributed SODIM system (SODIA).

  - By increasing the number of data sources, the execution time of test A increases compared to tests B and C, as they had larger number of cluster nodes and therefore, better job distribution than test A.

- *Test B:*
  - Test B gives its best execution time in the range (500–10,000) data sources compared to other tests, because it gave the best job distribution and the minimum cluster daemons overhead in that range of data sources.
  - Test B gives larger execution time compared to test A in the range (5–50) data sources because of the cluster daemons overhead that appears to be significant relative to the small application execution time and small number of data sources.
  - Test B gives larger execution time compared to test C starting from 40,000 data source, because test C had larger number of cluster nodes and therefore, it provides better job distribution.

- *Test C:*
  - Test C gives its best execution time for 40,000 data sources compared to other tests, because it provides the best job distribution and the minimum cluster daemons overhead in that range of data sources.
  - Test C gives larger execution time compared to test A and B in the range (5–10,000) data sources because of the effect of the cluster daemons overhead relative to small application execution time and small number of data sources.

4.3. SODIM versus non-distributed SODIM (SODIA)

Other interesting results from single node test and Cloud test are gathered and represented in this section for discussion.

As shown in Fig. 4, a comparison in execution time between SODIM (tested on Amazon EC2) and the non-distributed version (SODIA) is presented.

The execution time widely varies between the two systems, and SODIM running on Hadoop cluster gives remarkable results compared to non-distributed version showing that the distribution of integration tasks among mappers on different nodes reduces the execution time significantly especially in large data source numbers.

5. Conclusion and future extensions

The proposed SODIM system is the result of combining modern techniques, and technologies. The system adopted SOA as a design paradigm, web-service as a communication technology, and MapReduce execution model as the parallel processing framework with all special features it provides. By adopting SOA, SODIM system is able to address the requirements of loosely coupled, standards-based, and protocol-independent distributed computing. Using dynamic web-service invocation framework in system communications simplified system design. It also simplified the input and output transformations by adopting unified message model. SODIM is designed to work on a pool of commodity machines like those provided by Clouds and Grids and to process large numbers of data sources represented as web-services.

5.1. Future extensions

It is worthwhile investing an empirical model, relating number of machines and number of data sources in Hadoop cluster. A study shall be provided in applying fully automated discovery and binding techniques (such as semantic search) for related web-services. SODIM did not include studying security issues and implications on the system and how different security policies can be applied specially on distributed environments, and what modern infrastructures like Clouds can offer and support in this field.

References

[1] M. Papazoglou, W.-J. van den Heuvel, Service oriented architectures: approaches, technologies and research issues, The VLDB Journal 16 (2007) 389–415, http://dx.doi.org/10.1007/s00778-007-0044-3, <http://dx.doi.org/10.1007/s007 78-007-0044-3>.

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, C. Ferris, D. Orchard, Web service architecture, W3C, Working Group Note 11, Tech. Rep., 2004. <http://www.w3.org/TR/ws-arch/>.

[3] 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, Communications of the ACM 53 (4) (2010) 50–58, <http://doi.acm.org/10.1145/1721654.1721672>.

[4] C. Lam, Hadoop in Action, first ed., Manning Publications Co., Greenwich, CT, USA, 2010.

[5] The Apache Software Foundation, MapReduce tutorial, Tech. Rep., 2008.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom, The TSIMMIS approach to mediation: data models and languages, Journal of Intelligent Information Systems 8 (1997) 117–132, http://dx.doi.org/10.1023/A:1008683107812, <http://dx.doi.org/10.1023/A:1008683107812>.

[7] C. Batini, M. Lenzerini, S.B. Navathe, A comparative analysis of methodologies for database schema integration, ACM

Computing Surveys 18 (4) (1986) 323–364, <http://doi.acm.org/10.1145/27633.27634>.

[8] M.R. Genesereth, A.M. Keller, O.M. Duschka, Infomaster: an information integration system, SIGMOD Record 26 (2) (1997) 539–542, <http://doi.acm.org/10.1145/253262.253400>.

[9] A. Levy, A. Rajaraman, J. Ordille, Querying heterogeneous information sources using source descriptions, Stanford InfoLab, Tech. Rep., 1996–61, 1996. <http://ilpubs.stanford.edu:8090/191/>.

[10] M. Friedman, A. Levy, T. Millstein, Navigational plans for data integration, in: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, ser. AAAI '99/IAAI '99. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999, pp. 67–73. <http://dl.acm.org/citation.cfm?id=315149.315229>.

[11] L. Xu and D. Embley, Combining the best of global-as-view and local-as-view for data integration, Tech. Rep., 2004. <http://www.deg.byu.edu/papers/PODS.integration.pdf>.

[12] P. McBrien, A. Poulovassilis, Data integration by bi-directional schema transformation rules, in: Proceedings of the 19th International Conference on Data Engineering 2003, 2003, pp. 227–238.

[13] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro, Service-based software: the future for flexible software, in: Proceedings of the Seventh Asia–Pacific Software Engineering Conference, 2000. APSEC 2000, 2000, pp. 214 –221.

[14] F. Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgena, P. Breretona, J. Keane, P. Layzell, M. Rigby, J. Xu, Dynamic data integration using web services, in: Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 262–269.

[15] S. Sathya, M. Jose, Application of hadoop mapreduce technique to virtual database system design, in: 2011 International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), 2011, pp. 892 –896.

[16] P. Leitner, The daios framework dynamic asynchronous and message-oriented invocation of web services, Master's thesis, Vienna University of Technology, 2007.

[17] P. Leitner, F. Rosenberg, S. Dustdar, Daios: efficient dynamic web service invocation, IEEE Internet Computing 13 (3) (2009) 72–80.

[18] Amazon Web Services LLC, Amazon elastic mapreduce: Getting started guide, API Version 2009-03-31; Copyright 2012 Amazon Web Services LLC., Tech. Rep., 2009.

[19] Amazon Web Services LLC, Amazon elastic mapreduce: Developer guide, API Version 2009-03-31; Copyright 2012 Amazon Web Services LLC, Tech. Rep., 2009.

[20] Amazon Web Services LLC, Amazon elastic compute cloud: Developer guide, API Version 2009-03-31; Copyright 2012 Amazon Web Services LLC, Tech. Rep., 2009.