Big Data Processing by Volunteer Computing Supported by Intelligent Agents

Jerzy Balicki(✉), Waldemar Korłub, and Jacek Paluszak

Faculty of Telecommunications, Electronics and Informatics, Gdańsk University of Technology, Narutowicza St. 11/12, 80-233 Gdańsk, Poland
balicki@eti.pg.gda.pl, waldemar.korlub@pg.gda.pl, jacekpaluszak@gmail.com

Abstract. In this paper, volunteer computing systems have been proposed for big data processing. Moreover, intelligent agents have been developed to efficiency improvement of a grid middleware layer. In consequence, an intelligent volunteer grid has been equipped with agents that belong to five sets. The first one consists of some user tasks. Furthermore, two kinds of semi-intelligent tasks have been introduced to implement a middleware layer. Finally, two agents based on genetic programming as well as harmony search have been applied to optimize big data processing.

1 Introduction

Big data (an acronym BD) can be very useful to achieve high-value information related to decision support, business intelligence or forecasting. Large volumes of data are published by many companies to the web, and also they deploy e-commerce applications that enable continuous self-service transactions via the web. We observe a migration of database capacities from terabytes to petabytes. Furthermore, we can expect that modern systems will require distributed databases with exabytes or even zetabytes. It is difficult to define big data, e.g. 10 terabytes is a large capacity for a banking transaction system, but small even to test a web search engine. However, we can treat this data as big if a data capacity is large enough to be uncooperative to work with some relational database management systems RDBMS like DB2, INGRES, MySQL, *Oracle, Sybase* or SQL *Server* [28, 31].

Some crucial difficulties with big data are related to: capture, storage, search, sharing, analytics, and visualizing. Few exabytes of data are captured per day from different sources: sensors, GPS, smartphones, microphones, cameras, tablets, computer simulations, satellites, radiotelescopes, and social networks via some wireless sensor networks. In result, the data store capacity has approximately doubled every three years for thirty years. Furthermore, *The Internet of Things* supports BD gathering. Currently, we can expect to store more or less zetabytes. data storage, their visualization, analysis and search are still considered as an open problem to solve, too [14, 24].

BD is not convenient to the most RDBMS because massively parallel software on thousands of servers is required. In applications of statistics and visualization, sizes of BD exceed the capability of commonly used tools. A BD size can increase to achieve

J. Wang and C. Yap (Eds.): FAW 2015, LNCS 9130, pp. 268–278, 2015.
DOI: 10.1007/978-3-319-19941-2_26

many petabytes for one volume. Fortunately, progress in speed of data communication can support BD processing. Another feature of BD is wide variety of them, what is related to a huge range of data types and sources. So, the 4Vs model can be descripted by: high *volume*, extraordinary *velocity*, great data *variety*, and *veracity* [30]. In BD, some regressions can be used to find predictions. On the other hand, some descriptive statistics can be developed for business intelligence.

Big data processing requires high performance architecture of distributed systems. In this paper, volunteer computing systems are proposed for big data processing. In the volunteer grids such as BOINC, *Folding@home*, and GIMPS, flat data sets are transformed into several millions of subsets that are processed parallel by volunteer computers. Performance of grid computing as *Folding@home* is estimated at 40 [PFLOPS]. For comparison, the fastest supercomputer performance *Tianhe*-2 reached 34 [PFLOPS] in 2014 [8]. Moreover, the BOINC system performance is 6 [PFLOPS] and computing power achieved for the most important projects using this software are: *SETI@Home* – 681 [TFLOPS] and *Einstein@Home* – 492 [TFLOPS]. GIMPS with 173 [TFLOPS] discovered the 48th Mersenne prime in 2013. The number of active volunteers can be estimated as 238,000 for BOINC [8].

Moreover, intelligent agents can be developed to efficiency improvement of a grid middleware layer. For example, an experimental volunteer and grid system called *Comcute* that is developed at *Gdańsk University of Technology* can be equipped with agents that belong to five sets [5, 11]. This grid was a virtual laboratory for experiments with big data and intelligent agents. Especially, some user tasks like the *Collatz* hypothesis verification or the 49th *Mersenne'* prime finding can move autonomously with a big amount of data from some source databases to some destination computers, and then outcomes are returned. If we consider above tasks, a reduction of databases can be done by dynamic memorizing the current period of Integers. However, a dilemma appears if we study some simulations of fire spread that is another *Comcute* project. In that case, some scenarios are analyzed, and several strategies are found [6].

Furthermore, two kinds of intelligent tasks have been considered to implement a middleware layer. Agents for data management send data from source databases to distribution agents. Then, distribution agents cooperate with web computers to calculate results and return them to management agents. Both types of agents can autonomously move from one host to another to improve quality of grid resource using.

Moreover, two group of agents based on genetic programming as well as harmony search have been introduced to optimize big data processing. A set of agents designed for local optimization are some harmony search schedulers. These schedulers can optimize resource using. They cooperate with distributors and managers to give them information about optimal workload in a grid. Finally, genetic programming has been applied for finding the compromise configurations of grid. These agents cooperate with harmony search schedulers to correct in local timetables.

In this paper, big data dilemmas are described in Sect. 2. Then, intelligent agents for big data are studied in Sect. 3. A description of agents based on genetic programming is included in Sect. 4. Moreover, some outcomes for numerical experiments are submitted.

2 Big Data System Architectures

Some current big data applications are based on tools such as *Hadoop* or *NoSQL* cluster. Scalability is their ability to handle an increasing amount of transactions and stored data at the same amount of time. *MongoDB* is the *NoSQL* database that supports the data stored to different nodes and has support for a number of programming languages.

The current solutions to big data dilemmas are based on parallel data processing. We can use a massively parallel cluster with lots of CPUs, GPUs, RAM and disks to obtain a high performance by data-based parallelism. It is important to deal with OLTP *online* transaction processing that is a class of information systems to manage transaction-oriented applications. Moreover, low response time for decision-support queries can be obtained for OLAP that is online analytical processing to answering multi-dimensional analytical queries rapidly. High reliability can be obtained through data replication. As a final point, we expect extensibility with the almost linear speed-up as well as linear scale-up. Performance can increase linearly for a constant database size, load, and proportional increase of the components (CPU, GPU, RAM, disk). On the other hand, linear scale-up means that performance is constant for proportional increase of CPUs and a linear growth of load and database size.

Figure 1 shows three cases of data-based parallelism. In the first case, the same operation is carried out on different data (Fig. 1a). This case can be considered for large query. For concurrent and different queries that operate on the same data, we can consider the second case on Fig. 1b.

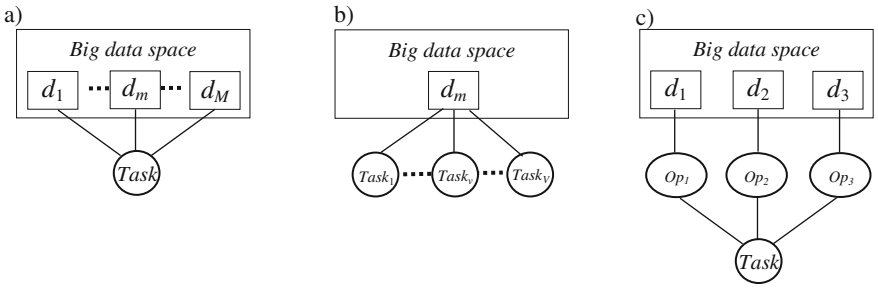

Fig. 1. Three cases of data-based parallelism

The third case (Fig. 1c) is related to complex query that is divided on some parallel operations acting on diverse data. Above three cases of data-based parallelism permit us to prepare two alternative architectures that support big data parallelism.

The main architecture that is convenient for write-intensive tasks is based on shared-memory computers like *Bull Novascale*, HP *Proliant* or IBM *Numascale*. Unfortunately, this architecture is based on NUMA *Non-Uniform Memory Access* server technology and it is not suited for big data. Several disks are shared by many processors via shared RAM. The architecture can support effectively applications, and it can support load balancing. On the other hand, the NUMA architecture is involved with interconnection limits and there are some difficulties with extensibility.

Architecture with shared-disk cluster is much more prepared for big data processing than NUMA architecture. *Storage Area Network* SAN interconnects private memory and disks that are shared by processors. Hosts like *Exadata, Oracle* RAC and IBM *PowerHA* are convenient for applications and some extensions can be made easily. However, a complex distributed lock manager is needed for cache coherence.

A crucial feature of BD is related to intensive reading from hard disks and then processing, instead of processing and then intensive writing. If we consider no sharing of memory or disks across nodes (Fig. 2), this system requires data partitioning of database like in servers: DB2 DPF, *MySQLcluster* or *Teradata*.

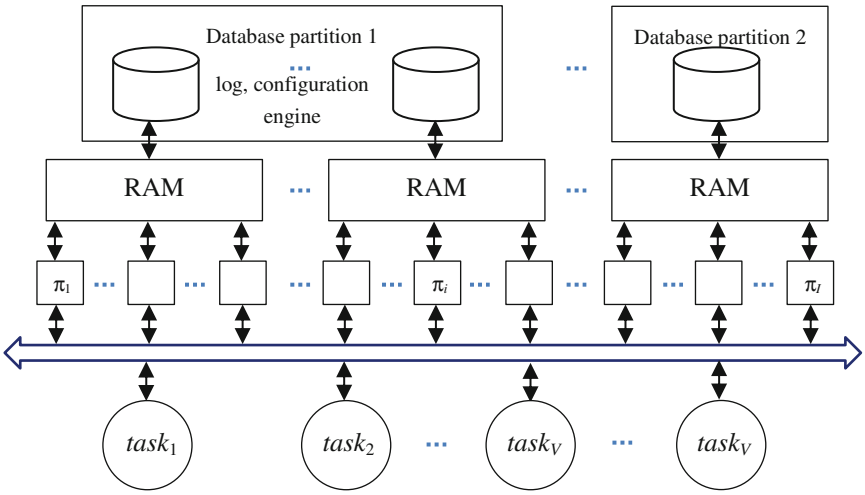

Fig. 2. Shared-nothing cluster architecture for big data [24]

Big data is spread over some partitions that run on one or some separate servers with own table spaces, logs, and configurations. A query is performed in parallel on all partitions. Such architecture can support *Google* search engine, *NoSQL* key-value stores (*Bigtable*). An advantage is the highest extensibility and low cost. In contrast, some updates and distributed transactions are not efficient.

3 Intelligent Agents for Big Data

Intelligent agents can improve efficiency of a grid middleware layer for big data processing. A volunteer grid can gather data from multiple sources, which may be heterogeneous and spread geographically across the world. Moreover, the collected data may be stored by a volunteer grid in multiple geographically spread facilities [9, 17].

Multiagent systems are well suited for BD acquisition because of mobility, which means the ability to move between different facilities. By doing that agents can get closer to the source of data or closer to the data they are about to process. It reduces bandwidth requirements and communication delays [9]. The ability to react upon sudden changes

of the environment and to act proactively is important to provide foundation for handling changes in availability of data sources or collected data. An agent can make decision if move to another set of data or initiate communication with other agents [2, 7].

Other useful traits of agents include abilities to communicate and negotiate. In agent-based data mining system it is possible to distinguish different roles and groups of tasks that constitute the whole mining process. Individual roles can be assigned to agents. Through communication and negotiation working groups of agents can be established, each of them built of agents with a unified goal. Agents can improve efficiency of data mining compared to centralized approaches [37]. It was applied in different domains showing promising results for further research, e.g. banking and finance domain [21] or resource allocation in distributed environments [4, 9].

A common approach for big data processing is the use of *MapReduce* algorithm, which is optimized for parallel and asynchronous execution on multiple computing nodes [12]. Because of the proven usefulness of this approach in multiple areas, e.g. bioinformatics [15, 26], fraud detection [22], social network analysis [16, 25] – there are many software frameworks for performing this kind of computations. Among most popular is the *Apache Hadoop* software [12], which can utilize computing power of multiple machines in a distributed environment.

However, such tools often introduce certain limitations. Of them is the need to use internal storage mechanisms (e.g. *Hadoop* distributed file system [29]), for effective operation. It is an effect of an inability to integrate with external and typically heterogeneous data sources. Data administrators are forced to move or duplicate large volumes of data from existing data stores to framework-specific ones. Another problem is the lack of support for online data and analysis. Moreover, many scenarios require extraction and merging of data to produce a meaningful result [32].

One more issue is that some frameworks may introduce architectural flaws like single point of failure (e.g. Hadoop before version 2.0 [33]) that have an influence on the whole system, in which they were deployed. Some of those issues can be addressed by using agent-oriented approach. Agents are designed for heterogeneous environments and are usually attributed with the ability to handle changes [36]. It translates to capability of integrating with different data sources. Reactive nature of agents enables them to work with online data streams with each new piece of information appearing in the stream interpreted as an event that requires agent response.

Another trait of agents is pro-activeness, which means that an agent can not only react to external events but also run actions on its own [34]. It is especially important in case of analysis, as there are often no clues about expected results known in advance. Because of that it is not possible to create the knowledge extraction algorithm *a priori*. Proactive behavior of agents can expose information that was not expected.

The problem of data acquisition can be solved be making use of another trait of agents – their mobility. It means that a software agent is not bound to any particular machine or execution container [20]. Individual agents can migrate between different nodes, to get closer to the sources of data. Instead of providing the data to the system, with multiagent approach the system acquires the information on its own so there is no need for data administrators to migrate or duplicate the data.

For real-time data analysis that takes into account both offline data and online streams Marz proposed the lambda architecture [23], which consists of three main components: batch layer, serving layer and speed layer. The batch layer is responsible for offline data processing and can be implemented using existing *MapReduce* frameworks like aforementioned *Hadoop*. This layer produces batch views of the data, which can be exposed to external applications. The serving layer serves prepared views to clients. The speed layer is responsible for real-time processing of data streams. It analyses data that was not yet processed by the batch layer. Speed layer produces real-time views that can be coupled with batch views to create complete representation of the extracted knowledge.

Twardowski and *Ryżko* further show that the lambda architecture can be defined in terms of a heterogeneous multiagent system [32]. There are several strong motivations for this approach. First of all, the lambda architecture gives only general guidelines. The actual realization requires integration of a few components: one for batch processing, another one for serving views, a different solution for real-time stream analysis and a component that merges real-time views with batch views. There are ready-made frameworks and libraries for each step that can be used to create a complete solution.

The use of multiagent environment will provide a common way for information exchange between different component and a common execution model [32]. The differences between individual components of the lambda architecture lead to inherently heterogeneous realizations so the ability to handle diversity in agent systems in another motivation for this approach. Figure 3 shows the lambda architecture employing multiagent model.

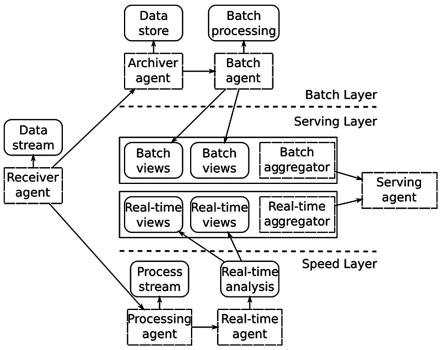

Fig. 3. Multiagent real-time processing utilizing lambda architecture [32]

Limitations of intelligent agents systems are related to higher complexity of software preparation that causes higher costs and higher probability of mistake appearing. What is more, some algorithms based on artificial intelligence have to be considered. These algorithms are probabilistic and it cause some unpredicted outcomes during some phases of their running. However, advantages of intelligent agents seem to be more important that their disadvantages. Among intelligent agents, agents based on genetic programming are worth to consideration regarding their capability of resource managing in grids applied for big data.

4 Agents Based on Genetic Programming

Intelligent agents based on genetic programming *AGPs* can optimize a grid resource management for big data queries [1, 18]. Especially, they have been dedicated to global optimization of middleware software module allocation in *Comcute* grid. In that system, they cooperate with intelligent agents based on harmony search *AHSs* that reconfigure some local parts of grids. *AGP* starts from a goal of load balancing to be achieved and then it creates a solver autonomously [19]. It is similar to deal with the dilemma from machine learning "How can computers be made to do what needs to be done, without being told exactly how to do it?" [27]. *AGP* uses the principle of selection, crossover and mutation to obtain a population of programs applied as a scheduler for efficient using big data by the *Comcute*. This scheduler optimizes some criteria related to load balancing and send a compromise solution to *AHSs* [35].

In *AGP*, a program is represented as a tree that consists on branches and nodes: a root node, a branch node, and a leaf node. A parent node is one which has at least one other node linked by a branch under it. The size of the parse tree is limited by the number of nodes or by the number of the tree levels. Nodes in the tree are divided on functional nodes and terminal ones. Mutation and crossover operate on trees, and chosen node from tree is a root of subtree that is modified regarding genetic operators.

Figure 4 shows architecture of the volunteer grid *Comcute* with one AGP agent that cooperates with two *AHS* agents. We divide the whole grid on several subgrids with at most 15 nodes.

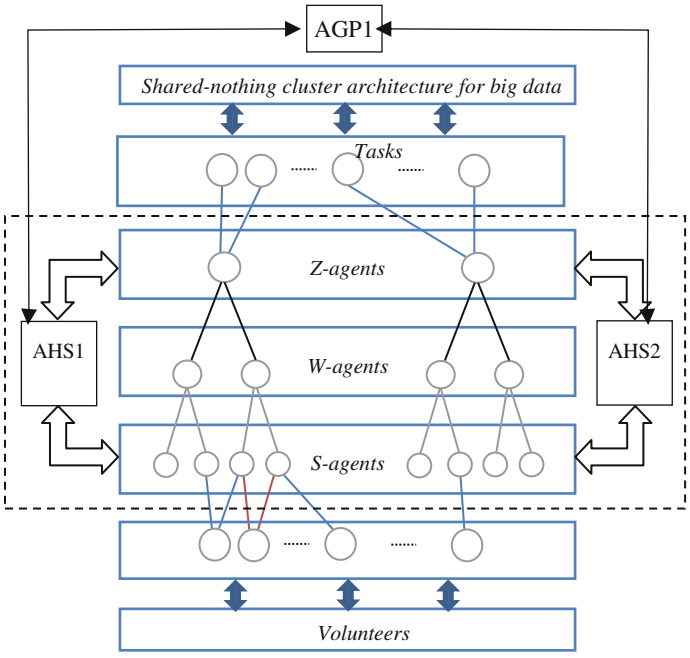

Fig. 4. Agent based on genetic programming in the *Comcute* for big data processing

AHS can find configuration for at most 15 nodes, 50 tasks and 15 alternatives of resource sets *ARS* on *PC/Windows 7/Intel i7*. Figure 5 shows an example of a compromise configuration for a subgrid in the *Comcute* that was found by *AHS1* for its area consisted of 15 nodes. Workload is characterized by two criteria [10, 13]. The first one is the CPU workload of the bottleneck computer (denoted as $\hat{Z}_{max}$), and the second one is the communication workload of the bottleneck server ($\tilde{Z}_{max}$).

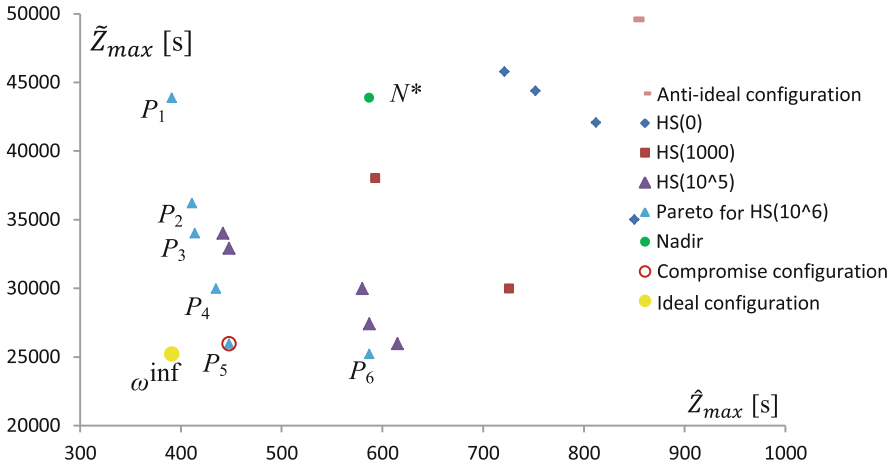

Fig. 5. A compromise configuration found by *AHS1*

The compromise configuration found by *AHS1* is specified in Table 1. The *W*-agent with number 5 as well as two *S*-agents (No. 6 and 27) should be moved to the node No. 1, where the third alternative of resource set *ARS* (*BizServer* E5-2660 v2) is assigned. Agent *AHS1* publishes this specification in the common global repository of grid and the other agents can read it to make decision related to moving to some recommended nodes. However, reconfiguration of resource requires a bit of time. So, a middleware agent reads the state of resources in the given grid node, and then it makes decision whether to go to that node or not.

Table 1. A specification of a compromise configuration from AHS1

Node i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ARS j	3	8	3	9	8	9	8	3	9	3	3	3	3	3
No. W	5		11	4	9		2,13	10		1,8	6,15	3,7	14	12
No. S	6,27	13,23,24,30	7,26	14,21,28	3,15,25	4,5,11,22	19	2	1,16,18,20	9	10	29	12,17	8

On the other hand, the *AGP* cooperates with several *AHSs*. It takes into account their recommendation for resource using by middleware agents. Moreover, the *AGP* optimizes the resource usage for the whole grid starting from the configuration obtained by set of *AHSs* and trying to improve it by multi-criteria genetic programing.

5 Concluding Remarks and Future Work

Shared-nothing cluster architecture for big data can be extended by cooperation with volunteer and grid computing. Moreover, intelligent agents in the middleware of grid can significantly support efficiency of proposed approach. Multi-objective genetic programming as relatively new paradigm of artificial intelligence can be used for finding Pareto-optimal configuration of the grid. Agents based on genetic programing can cooperate with harmony search agents to solve NP-hard optimization problem of grid resource using.

Our future works will focus on testing the other sets of procedures and terminals to find the compromise configurations for different criteria. Initial numerical experiments confirmed that sub-optimal in Pareto sense configurations can be found by *AGPs* and *AHSs*. Moreover, quantum-inspired algorithm can support big data, too [3].

Acknowledgements. This research is supported by Department of Computer Architecture, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology under statutory activity grant.

References

1. Altameem, T., Amoon, M.: An agent-based approach for dynamic adjustment of scheduled jobs in computational grids. J. Comput. Syst. Sci. Int. **49**, 765–772 (2010)
2. Balicki, J.: Negative selection with ranking procedure in tabu-based multi-criterion evolutionary algorithm for task assignment. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3993, pp. 863–870. Springer, Heidelberg (2006)
3. Balicki, J.: An adaptive quantum-based multiobjective evolutionary algorithm for efficient task assignment in distributed systems. In: Mastorakis, N., et al. (eds.) Proceedings of the 13th WSEAS International Conference on Computers, Recent Advances in Computer Engineering, Rhodes, Greece, pp. 417–422 (2009)
4. Balicki, J., Kitowski, Z.: Multicriteria evolutionary algorithm with tabu search for task assignment. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 373–384. Springer, Heidelberg (2001)
5. Balicki, J., Korłub, W., Szymanski, J., Zakidalski, M.: Big data paradigm developed in volunteer grid system with genetic programming scheduler. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part I. LNCS, vol. 8467, pp. 771–782. Springer, Heidelberg (2014)
6. Balicki, J., Korlub, W., Krawczyk, H., et al.: Genetic programming with negative selection for volunteer computing system optimization. In: Paja, W.A., Wilamowski, B.M. (eds.) Proceedings of the 6th International Conference on Human System Interactions, Gdańsk, Poland, pp. 271–278 (2013)

7. Bernaschi, M., Castiglione, F., Succi, S.: A high performance simulator of the immune system. Future Gener. Comput. Syst. **15**, 333–342 (2006)
8. BOINC. http://boinc.berkeley.edu/, Accessed 25 Jan 2015
9. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent mining: the synergy of agents and data mining. IEEE Intell. Syst. **24**, 64–72 (2009)
10. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002)
11. Comcute grid. http://comcute.eti.pg.gda.pl/, Accessed 25 Jan 2015
12. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**, 1–13 (2008)
13. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
14. Finkelstein, A., Gryce, C., Lewis-Bowen, J.: Relating requirements and architectures: a study of data-grids. J. Grid Comput. **2**, 207–222 (2004)
15. Gunarathne, T., et al.: Cloud computing paradigms for pleasingly parallel biomedical applications. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, pp. 460–469 (2010)
16. Guojun, L., Ming, Z., Fei, Y.: Large-scale social network analysis based on MapReduce. In: Proceedings of the International Conference on Computational Aspects of Social Networks, pp. 487–490 (2010)
17. Jennings, N.R., Wooldridge, M.: Applications of intelligent agents. In: Jennings, N.R., Wooldridge, M. (eds.) Intelligent Agents, pp. 3–28. Springer, New York (1998)
18. Kang, J., Sim, K.M.: A multiagent brokering protocol for supporting Grid resource discovery. Appl. Intell. **37**, 527–542 (2012)
19. Koza, J.R., et al.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, New York (2003)
20. Leyton-Brown, K., Shoham, Y.: Multiagent Systems: Algorithmic: Game-theoretic and Logical Foundations. Cambridge University Press, Cambridge (2008)
21. Li, H.X., Chosler, R.: Application of multilayered multi-agent data mining architecture to bank domain. In: Proceedings of the International Conference on Wireless Communications and Mobile Computing, pp. 6721–6724 (2007)
22. Mardani, S., Akbari, M.K., Sharifian, S.: Fraud detection in process aware information systems using MapReduce. In: Proceedings on Information and Knowledge Technology, pp. 88–91(2014)
23. Marz, N., Warren, J.: Big Data - Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications Co., USA (2014)
24. O'Leary, D.E.: Artificial intelligence and big data. IEEE Intell. Syst. **28**, 96–99 (2013)
25. Ostrowski, D.A.: MapReduce design patterns for social networking analysis. In: Proceedings of International Conference on Semantic Computing, pp. 316–319 (2014)
26. Qiu, X., et al.: Using MapReduce technologies in bioinformatics and medical informatics. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Portland (2009)
27. Samuel, A.L.: Programming computers to play games. Adv. Comput. **1**, 165–192 (1960)
28. Shibata, T., Choi, S., Taura, K.: File-access patterns of data-intensive workflow applications. In: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 522–525 (2010)
29. Shvachko, K., et al.: The Hadoop distributed file system. In: MSST, pp. 1–10 (2010)
30. Snijders, C., Matzat, U., Reips, U.-D.: 'Big Data': big gaps of knowledge in the field of Internet. Int. J. Internet Sci. **7**, 1–5 (2012)

31. Szabo, C., et al.: Science in the cloud: allocation and execution of data-intensive scientific workflows. J. Grid Comput. **12**, 223–233 (2013)
32. Twardowski, B., Ryzko, D.: Multi-agent architecture for real-time big data processing. In: Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technologies, vol. 3, pp. 333–337 (2014)
33. Vavilapalli, V.K.: Apache Hadoop yarn: Yet another resource negotiator. In: Procedings of the 4th Annual Symposium on Cloud Computing, New York, USA, pp. 5:1–5:16 (2013)
34. Verbrugge, T., Dunin-Kęplicz, B.: Teamwork in Multi-Agent Systems: A Formal Approach. Wiley, Chichester (2010)
35. Węglarz, J., Błażewicz, J., Kovalyov, M.: Preemptable malleable task scheduling problem. IEEE Trans. Comput. **55**, 486–490 (2006)
36. Wooldridge, M.: Introduction to Multiagent Systems. Wiley, Chichester (2002)
37. Zhou, D., et al.: Multi-agent distributed data mining model based on algorithm analysis and task prediction. In: Proceedings of the 2nd International Conference on Information Engineering and Computer Science, pp. 1–4 (2010)