Paper

Towards an Agent-Based Augmented Cloud

Roman Dębski, Aleksander Byrski, and Marek Kisiel-Dorohinicki

Department of Computer Science, AGH University of Science and Technology, Kraków, Poland

Abstract—In the paper an agent-based framework deployed in hybrid cluster and volunteer computing environment is presented. It utilizes two concepts proposed by the authors: *Augmented Cloud* and *Agent Platform as a Service (AgPaaS)*. Both concepts are discussed in the context of *Cloud Computing* as defined by NIST. The key idea of the presented solution is to span the cloud (i.e., computing infrastructure) beyond the data center borders by utilizing web browsers as computational workers. The feasibility of the approach was demonstrated by two prototypes: the first one was based on Java Applets and Adobe Flash, whereas the second one on Microsoft Silverlight. The prototypes were next used to perform simple experiments, mainly related to scalability issues. Selected results from the experiments are discussed in the final part of the paper.

Keywords—Agent Platform as a Service, Augmented Cloud, Cloud Computing, Multi-Agent Systems.

1. Introduction

Many large-scale[1] computational problems can be effectively solved in distributed environments which are based on the agent paradigm. Some examples are: marketplace simulations, modeling and control of autonomous robots and many optimization problems (complex multi-criteria, multi-modal and/or the ones in which the evaluation of a single solution is simulation based). In many of these cases *scalability* is the central issue. Both from the point of view of software (algorithms, logical architecture) and hardware (physical architecture).

Very often the hardware infrastructure scalability is the real issue. In the case of possible non-deterministic changes in the computation power demands[2], caused, e.g., by certain parameters of the computation model (cf. experiments with computational multi-agent systems presented in [1]), dynamic adaptation of the computation load is required (e.g., load balancing with diffusion-based scheduling as described in [2]). These problems are usually addressed by the choice of some cloud-oriented solutions [3]. Yet the problem of limited resources and the costs of maintaining often wasted computing power still remains open.

In the paper a new, cost effective, approach to building a highly scalable execution environment, particularly dedicated for computational systems which may be realized in agent-based paradigm, is presented. It utilizes two concepts proposed by the authors in [4]: *Augmented Cloud* and *Agent Platform as a Service (AgPaaS)*. Both concepts are discussed in the context of *Cloud Computing* as defined by NIST [5]. The key idea of the presented solution is to span the cloud (i.e., computing infrastructure) beyond the data center borders by utilizing web browsers as computational workers.

In the first section *AgE* – as an example of an agent-based computation platform (framework) is described. Next, its web browser-based implementation is presented. In the subsequent part, the concepts of *Augmented Cloud* and *Agent Platform as a Service* are defined. Since the contribution is a direct follow-up of [4], experimental results extending the ones presented there, constitute the final part of the paper.

2. AgE – Agent-Based Computation Platform

AgE platform[3] is designed to support building a wide range of agent-based optimization and simulation systems utilizing various meta-heuristics such as evolutionary algorithms. A *computation task* to be executed on the AgE platform is defined by providing a *computation description file*, which includes (among other things): the computation decomposition details (i.e., types of agents, their structure, types of operations that specify algorithms), problem-dependent parameters and the problem stop condition. Based on the description file, on the platform start-up, the computation context is built from the necessary components. Next, all the required agents and their environments are created, configured and distributed among nodes. Finally, the computation can be started and is performed until it reaches the stop condition. During the execution time some of the computational agents are attached to the platform monitors. The monitors are responsible for collecting problem-dependent data, used to visualize the current state and the final results of the computation.

The computation task is decomposed and the sub-tasks are assigned to the computational agents. The agents are structured into a tree (as shown in Fig. 1), according to the algorithm decomposition (it is worth to notice that each aggregate is also an agent). It is assumed, that all the sub-

[1]The term "large-scale problem" is used here in a broad sense, referring to all problems considered difficult for all known solution methods.

[2]When the demand for computation power is deterministic and can be reasonably estimated, a straightforward solution would be to use a computer cluster with well-defined, still limited resources.

[3]AgE is developed at AGH University of Science and Technology.

tasks assigned to the agents at the same level are executed in parallel. To increase performance, the top level agents (called *workplaces*) along with all their children can be distributed amongst many nodes.

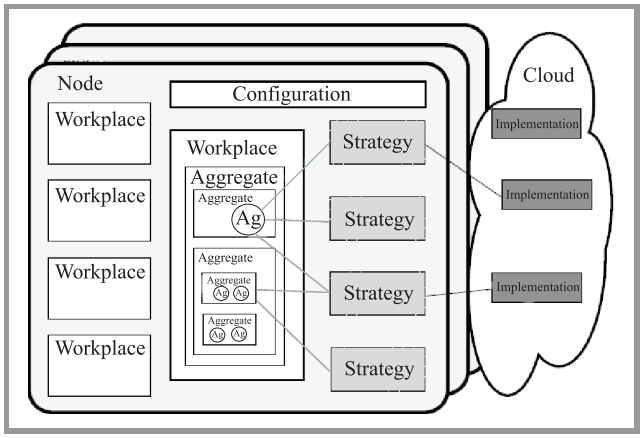

Fig. 1. Agent tree structure.

Agents, however, are not atomic assembly units, but they can be further decomposed into functional units according to *Strategy* design pattern [6]. Strategies represent problem-dependent algorithm operators and may be exchanged without intruding the agents' implementation. Their instances may be shared between agents as they provide various services to agents or others strategies.

Any part of a single agent's task can be delegated according to *Strategy* design pattern (see Fig. 1). Strategies can represent problem-dependent algorithm operators (mutation, evaluation of fitness, etc.) and may be exchanged without intruding agents' implementation. The delegated operations can be executed by external resources – this approach can be considered as a recommended way of the AgE platform expansion, especially in case of operations with a low communication-to-computation ratio. Fitness evaluation of a single solution can be given as an example of such delegated task (this approach was used in both prototypes discussed in the final section). In this case the computation can be based on the master-slave model [7], and the slaves (as fitness evaluators) can be: nodes connected in computation clusters dedicated to particular task, volunteers [8] (also web browser based [9]), or sideband computing applications [10].

3. Extending the AgE: A Web Browsers Based Approach

The main goal of the presented solution was to provide high scalability while remaining cost-effective. It has been achieved by utilizing web browsers as computational workers. The architecture of the platform is shown in Fig. 2. It is comprised of three layers:

- *The agent-based environment layer* provides a hierarchy of agents responsible for the realization of a given meta-heuristic. During their work, they generate tasks, which are delegated, using *Proxy* design pattern, to a connector that than passes it further to the application servers layer. Tasks are realized as asynchronous operations, so that agents can continue their work without waiting for results, unless they are required for further processing. Communication between the environment and the application servers layer is realized by sending requests, which identify tasks to be performed and input data.

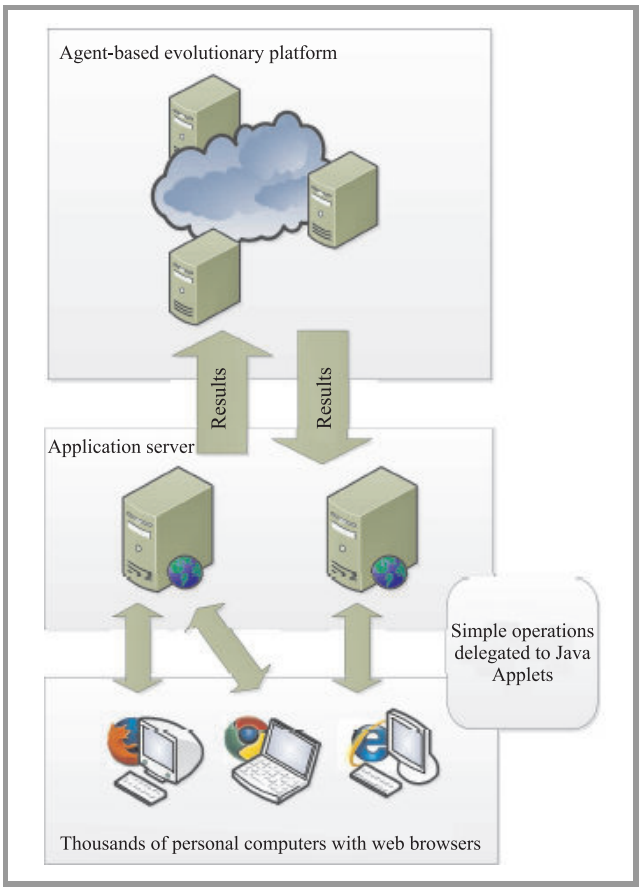

Fig. 2. Architecture overview.

- *The application servers layer* dispatches ordered tasks amongst available computational workers and passes back the results to the ordering agents. The number of workers depends on the number of the users whose web browsers cooperate with the platform (i.e., users that visit the services associated with the platform). It makes the workers' environment very dynamic, with the fault tolerance as the central issue to be addressed[4].

- *The web browser layer* utilizes applets[5] as computational workers which are responsible for executing the ordered tasks and returning the results to

[4]Consider users disconnecting the system in random moments.
[5]Programs executed in the context of a web browser, not necessary Java applets.

the application server. The applets are downloaded by browsers when a user visit any site connected to the system.

The approach can be applied both to building new computing environments and to extending the existing ones. However, one has to remember that the solution is suitable only for coarse grained problems (i.e., for which computation-to-communication ratio is high). Otherwise, the gained computing resources could be completely reduced by the communication overhead.
One can generalize the approach described here – it leads to the concept of *Augmented Cloud* [4], which is discussed in the next section.

4. Agent Platform in Augmented Cloud

An agent-based platform [11] dedicated for large-scale computations has to be designed for high scalability and deployed on a highly scalable execution environment like supercomputer, cluster, grid or cloud. Of the four, the last one is becoming more and more popular (also in High Performance Computing, e.g., [12], [13]) mainly because it is (at least can be) very cost effective [14], [15].
The approach discussed below is a hybrid of a classical cluster solution (as a backbone; can be considered as a classic cloud which spans the computers in a single data center), augmented with a web browsers based environment of volunteers, acting together as an *Augmented Cloud* [4].

4.1. Cloud Computing

A dedicated runtime environment (e.g., supercomputer, cluster) has many advantages. Yet, its scalability is restricted by the available computing resources (i.e., number of processors in a supercomputer or number of nodes in the cluster). That is why nowadays we observe a migration of services, software or even a whole computing infrastructure into "clouds" (like Windows Azure[6], Amazon EC2[7], or Google App Engine).
At this point it is worth to define *Cloud Computing* as the term has many definitions (e.g., [3] , [5], [15]) and there seems to be no consensus on what (precisely) a *Cloud* and *Cloud Computing* are [3]. According to NIST [5] *Cloud Computing* is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The essential characteristics are: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service.

[6]http://www.microsoft.com/windowsazure/
[7]http://aws.amazon.com/ec2/

The service models defined by NIST [5] are as follows:

- Software as a Service (SaaS) – the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based e-mail). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.
- Platform as a Service (PaaS) – the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.
- Infrastructure as a Service (IaaS) – the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

IBM in [16] introduces also the fourth service model – BPaaS:

- Business Process as a Service (BPaaS) – any business process (horizontal or vertical) delivered through the cloud service model (Multi-tenant, self-service provisioning, elastic scaling and usage metering or pricing) via the Internet with access via web-centric interfaces and exploiting web-oriented cloud architecture. The BPaaS provider is responsible for the related business function(s).

4.2. Augmented Cloud – Agent Platform as a Service

Combining the concepts of *the cloud* (as a way of providing the illusion of infinite computing resources available on demand) and the *web browser-based volunteer computing* [9] (as a way of collecting computational resources) leads to a practically zero-cost and highly scalable solution, which may be called an *Augmented Cloud* [4]. The scalability of the classical cloud is limited by the number of nodes in the data center on which the cloud is based/deployed; the proposed solution *augments* the classical cloud by a significant increase of its scalability.

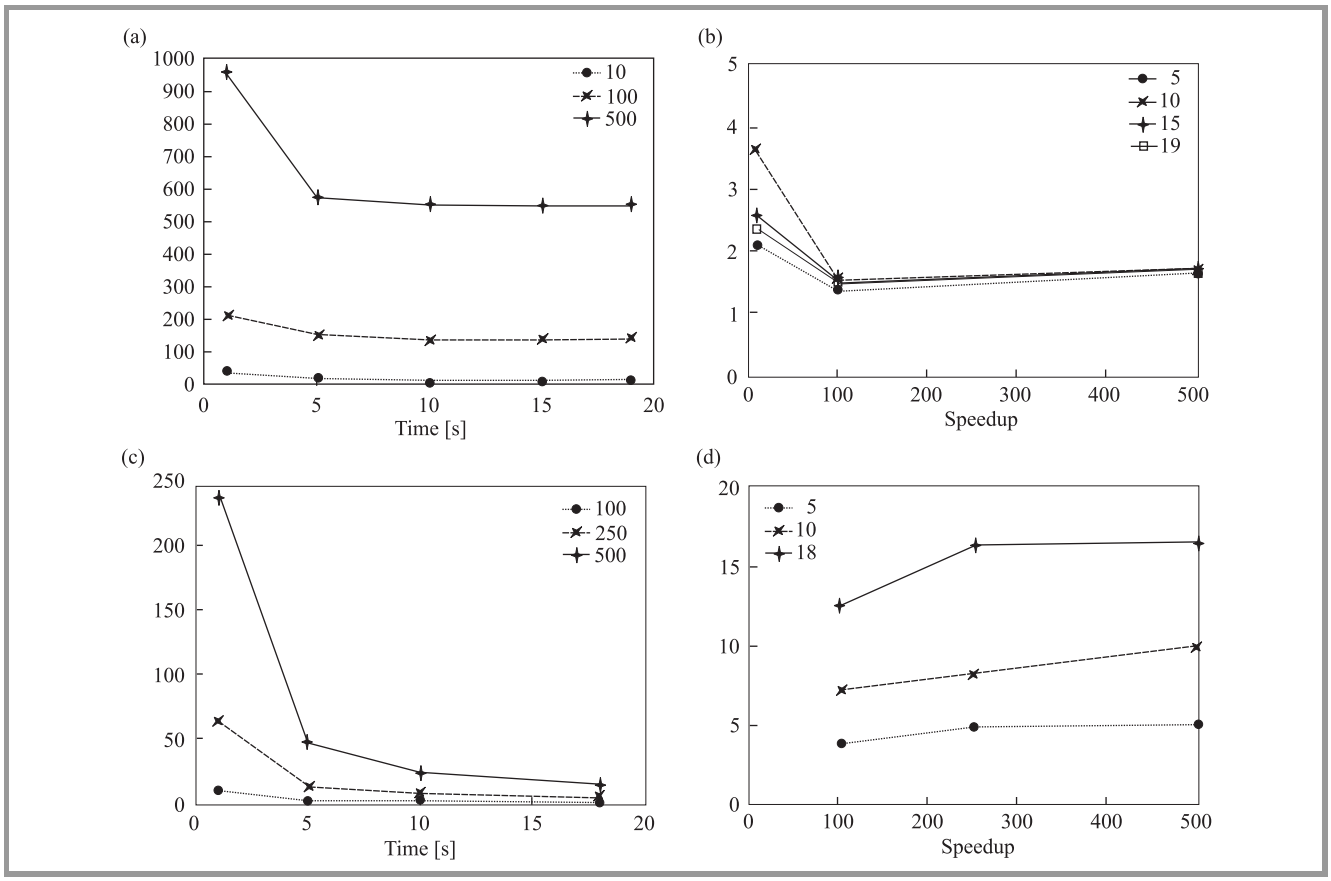

Fig. 3. Computation time [s] and speedup dependent on population sizes for different number of clients: (a), (b) – Java Applets; (c), (d) – Adobe Flash.

From the cloud computing service models [5] point of view the *Augmented Cloud* can be seen as a kind of Infrastructure as a Service (IaaS) in which (at least some of) the computational resources are web-browsers based. So it can be just a base for the target platform mostly because this level of abstraction is too low when taking into account both the development and deployment of (multi)agent-based systems.

What is needed is a set of services/libraries forming a highly scalable(software) platform dedicated for the target domain – agent-based systems. It can be considered as the second layer of the Cloud Computing Reference Architecture [16] which corresponds to the NIST Platform as a Service (PaaS) [5]. In this context it can be named *Agent Platform as a Service (AgPaaS)* to emphasize the agent-orientation of the platform [4].

5. Experimental results

In order to evaluate the concept of *Augmented Cloud*, computing time and speedup has been tested for a classical master-slave model of a parallel evolutionary algorithm, in which the computation of fitness values was delegated to browser-based slaves (as described in the previous sections). The experiments were based on two prototypes: in the first one the web browser layer was based on Java Applets and Adobe Flash, and in the second one on Microsoft Silverlight. The results are described in the next two subsections.

5.1. Java Applet and Adobe Flash

In this case the computing environment had a two-tier/layer architecture and consisted of a backbone server running a master process and two types of clients (slaves): Java Applet based (communicating with the server using Java RMI) and the Adobe Flash based. The algorithm was solving a typical benchmark optimization problem (Rastrigin function) for different population sizes and different number of clients.

According to expectations, increasing the number of slaves decreases the overall computation time (see Fig. 3). This effect is strongly dependent on the technology: the Java Applet slaves (see Fig. 3(a)) are (much) less effective than the Adobe Flash ones (see Fig. 3(c)). It is mainly because of the Java RMI communication overhead.

Figures 3(b) and 3(d) show the same data but from the computation speedup perspective. One can easily notice that in case of Java Applets there is practically no speedup. It is a good visualization of the importance of computation-to-communication ratio.

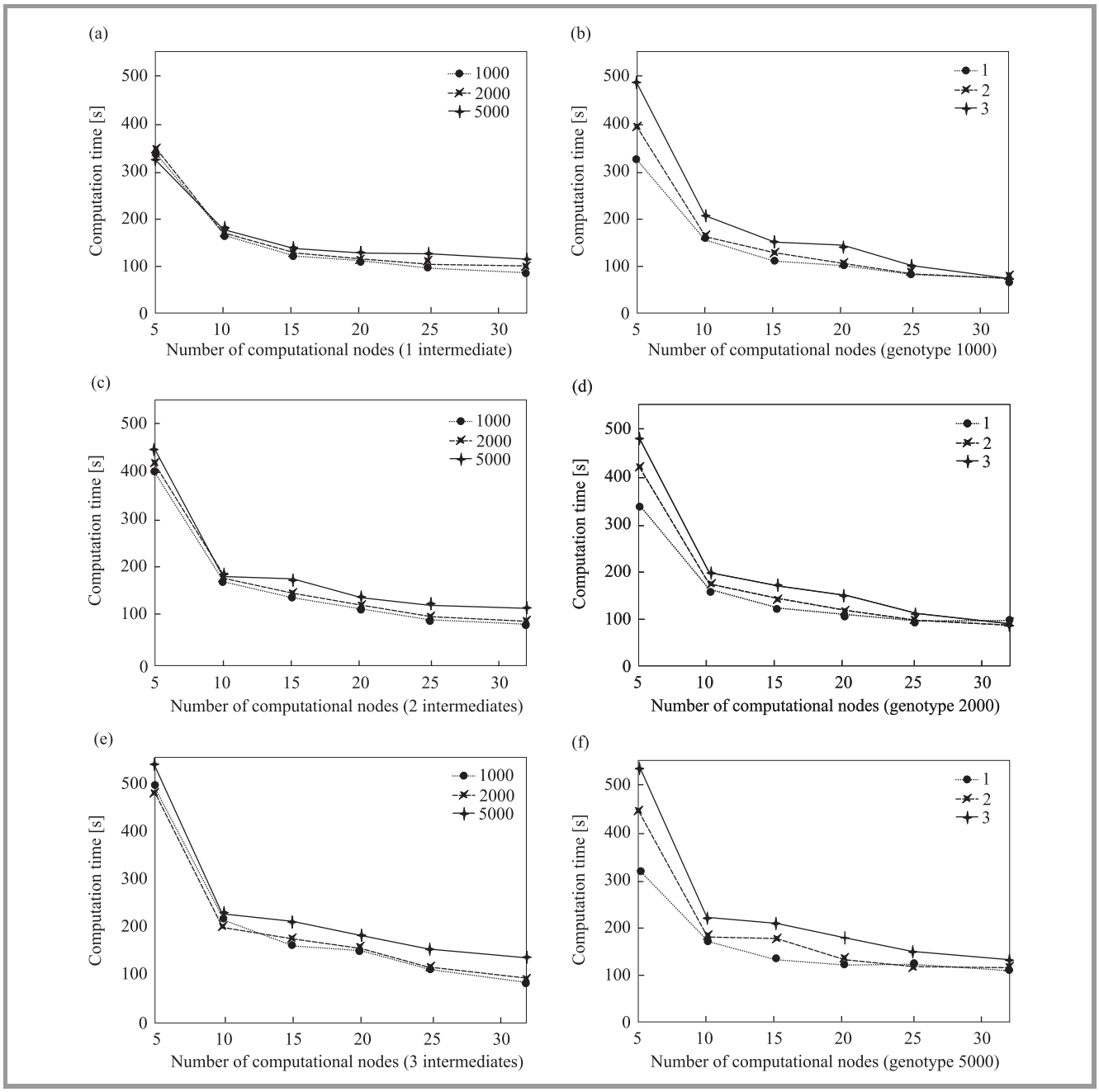

Fig. 4. In the left column: computation time as a function of the number of computational nodes, presented for different numbers of intermediate nodes and for different genotype lengths, in the right column: computation time as a function of the number of computational nodes, presented for different lengths of genotype and for different number of intermediate nodes.

5.2. Microsoft Silverlight

In this experiment the computing environment had a hierarchical[8], three-tier/layer structure.

- The backbone server (the master) – it was executing a simple genetic algorithm, delegating the calculation of each individual's fitness to web browser-based computational workers (slaves) via the application controllers layer.
- Application controllers (intermediate layer) – each controller is responsible for the pool of computational workers allocated to it; this layer can be utilized in different ways, e.g., to improve the system availability or/and reliability, to support load balancing, to control the value of computation-to-communication ratio (note: in the prototype none of the above was implemented).

[8]The environment can be visualized as a tree: level 1 – the backbone server (as the root), level 3 – the computational workers (as leaves), level 2 – controllers of sub-trees.

- Microsoft Silverlight based computational workers.

  Note: it can be noticed that the computing environment is similar to the one presented in Fig. 2, but with one exception: the AgE platform has been reduced to a single master process, which executed a simple genetic algorithm.

Figures 4(a), 4(c) and 4(e) show the computation time as a function of the number of computational nodes, for different numbers of intermediate servers (application controllers). One can notice that in the context of this computation, the introduction of intermediate nodes caused a significant overhead (in the whole range of analyzed computational node numbers), and in consequence, worsened the computation time. But as the number of the computational nodes was increasing this overhead was becoming smaller and smaller (close to zero for the last point of the curve). So as in case of most distributed systems, the performance of this one is strongly dependent on its configuration.

Figures 4(b), 4(d), and 4(f) present the same data from a different perspective.

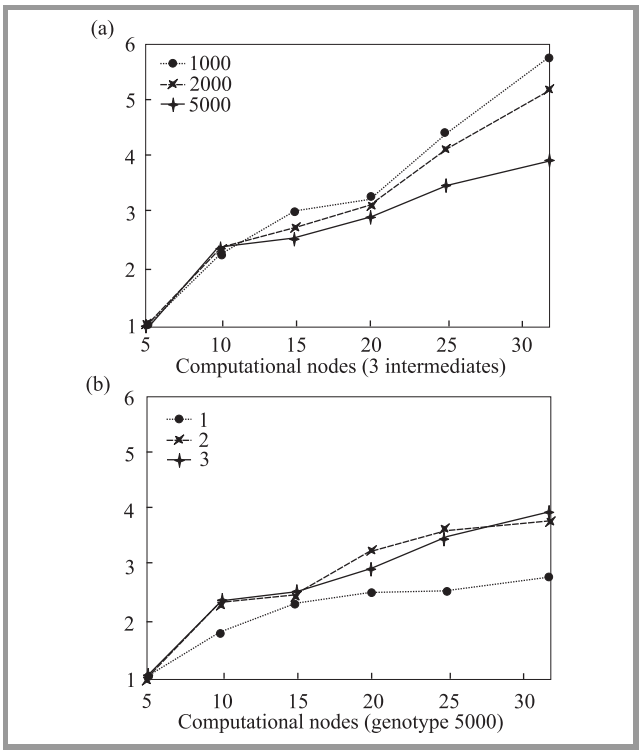

Fig. 5. Speedup as a function of the number of computational nodes: (a) in the configuration with 3 intermediate nodes, presented for different genotype lengths; (b) with the genotype length set to 5000, presented for different number of intermediate nodes.

Figures 5(a) and 5(b) present the system performance from the speedup point of view. An interesting effect is shown in Fig. 5(a): the smaller the genotype length, the bigger the speedup (and this effect is stronger as the number of computational nodes is increasing). This is again caused by the communication overhead: the bigger the genotype length, the more time is needed by the intermediate node to transfer its data. When a controller (intermediate node) handles many computational nodes its processing is I/O bound and, in consequence, it can become a bottle-neck of the whole system.

6. Conclusion

In the course of the contribution, after describing the AgE (as an example of an agent-based computation platform), and a possible way of its extension, the concepts of Augmented Cloud and the Agent Platform as a Service were introduced as way to address the scalability issues, which are present in many large-scale computations performed in the agent-based distributed environments.

In the final part of the paper, selected results obtained for a proof of concept kind of prototypes were shown. The computing environment of the first prototype consisted of a backbone server (master) and the two types of clients (slaves): Java Applet and Adobe Flash ones. The second prototype had a three-tier architecture with the computational workers (clients/slaves) based on Microsoft Silverlight.

The obtained results encourage further research and broader implementation of the Augmented Cloud concept, at the same time bringing more awareness about affecting the potential results by choosing appropriate technology for the implementation both of the server and web-browser clients, as well as for the communication.

In the near future the authors plan to conduct broader experiments with volunteer nodes based on different web technologies (e.g., JavaScript/WebWorkers). A long-term goal is to further evaluate the concept of Agent Platform as a Service (AgPaaS).

Acknowledgements

The work presented in this paper was partially supported by the Polish National Center of Research and Development grant No. 0108/R/T00/2010/11.

References

[1] A. Byrski, M. Kisiel-Dorohinicki, and M. Carvalho, "A crisis management approach to mission survivability in computational multi-agent systems", *Comp. Science*, vol. 11, pp. 99–113, 2010.

[2] M. Grochowski, R. Schaefer, and P. Uhruski, "Diffusion based scheduling in the agent-oriented computing system", in *Proc. Int. Conf. PPAM 2003*. Springer, 2004.

[3] S. Lu, I. Foster, and I. Raicu, "Cloud computing and grid computing 360-degree compared", in *Grid Comput. Environments Worsh. GCE'08*, Austin, Texas, USA, 2008.

[4] A. Byrski, R. Debski, and M. Kisiel-Dorohinicki, "Agent-based computing in augmented cloud environment", *Int. J. Comp. Sys. Sci. Eng.*, 2012 (accepted for printing).

[5] P. Mell and T. Grance, "The nist definition of cloud computing (draft)", Tech. rep., National Institute of Standards and Technology, 2011.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[7] E. Cantú-Paz, "A summary of research on parallel genetic algorithms", IlliGAL Rep. No. 95007, University of Illinois, 1995.

[8] D. P. Anderson, E. Korpela, and R. Walton, "High-performance task distribution for volunteer computing", in *Proc. 1st Int. Conf. e-Sci. Grid Comput.*, Melbourne, Australia, 2005.

[9] L. F. G. Sarmenta and S. Hirano, "Bayanihan: Building and studying web-based volunteer computing systems using Java", *Future Gener. Comp. Systems*, vol. 15, pp. 675–686, 1999.

[10] Y. Xu, "Global sideband service distributed computing method", in *Proc. Int. Conf. Commun. Netw. Distrib. Sys. Model. Simul. CNDS'98*, San Diego, CA, USA, 1998.

[11] B. Czerwinski, R. Debski, and K. Pietak, "Distributed agent-based platform for large-scale evolutionary computations", in *Proc. 5th Int. Conf. Complex Intel. Softw. Intensive Sys.*, IEEE Press, 2011.

[12] K. R. Jackson *et al.*, "Performance analysis of high performance computing applications on the amazon web services cloud", in *Proc. 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom'10*, Indianapolis, USA, 2010.

[13] J. Napper and P. Bientinesi, "Can cloud computing reach the top500", in *Proc. UCHPC-MAW '09*, ACM, 2009.

[14] J. J. Merelo *et al.*, "Browser-based distributed evolutionary computation: performance and scaling behavior", in *Proc. 9th Ann. Conf. Genetic Evol. Comput. GECCO '07*, New York, NY, USA, 2007, pp. 2851–2858.

[15] M. Armbrust *et al.*, "Above the clouds: A Berkeley view of cloud computing", Tech. rep., UC Berkeley, 2009.

[16] M. Behrendt *et al.*, "IBM cloud computing reference architecture 2.0", Tech. rep., The Open Group, 2011.

Roman Dębski holds the M.Sc. in Computer Science (AGH University of Science and Technology) and in Mechanical Engineering and the Ph.D. in Computational Mechanics (both obtained at Cracow University of Technology). He is an independent consultant with over 14 years of experience in IT. His current research focuses on the cloud computing.

E-mail: Roman.J.Debski@googlemail.com
Deptartment of Computer
AGH University of Science and Technology
Mickiewicza Av. 30
30-059 Kraków, Poland

Aleksander Byrski obtained his Ph.D. in 2007 at AGH University of Science and Technology in Kraków. He works as an assistant professor at the Department of Computer Science of AGH-UST. His research focuses on multi-agent systems, biologically-inspired computing and other soft computing methods.

E-mail: Olekb@agh.edu.pl
Deptartment of Computer Science
AGH University of Science and Technology
Mickiewicza Av. 30
30-059 Kraków, Poland

Marek Kisiel-Dorohinicki obtained his Ph.D. in 2001 at AGH University of Science and Technology in Kraków. He works as an assistant professor at the Department of Computer Science of AGH-UST. His research focuses on intelligent software systems, particularly utilizing agent technology and evolutionary algorithms, but also other soft computing techniques.

E-mail: Doroh@agh.edu.pl
Deptartment of Computer Science
AGH University of Science and Technology
Mickiewicza Av. 30
30-059 Kraków, Poland