The Multi-level Adaptive Approach for Efficient Execution of Multi-scale Distributed Applications with Dynamic Workload

Denis Nasonov[1](✉), Nikolay Butakov[1], Michael Melnik[1], Alexandr Visheratin[1], Alexey Linev[2], Pavel Shvets[3], Sergey Sobolev[4], and Ksenia Mukhina[1]

[1] ITMO University, Saint-Petersburg, Russia
denis.nasonov@gmail.com, alipoov.nb@gmail.com, mihail.melnik.ifmo@gmail.com, mukhinaks@gmail.com

[2] Lobachevsky State University of Nizhni Novgorod, Nizhny Novgorod, Russia
alipoov.nb@gmail.com

[3] Research Computing Center of Moscow State University, Moscow, Russia
shvets.pavel.srcc@gmail.com

[4] Moscow State University, Moscow, Russia
sergeys@parallel.ru

Abstract. Today advanced research is based on complex simulations which require a lot of computational resources that usually are organized in a very complicated way from technical part of the view. It means that a scientist from physics, biology or even sociology should struggle with all technical issues on the way of building distributed multi-scale application supported by a stack of specific technologies on high-performance clusters. As the result, created applications have partly implemented logic and are extremely inefficient in execution. In this paper, we present an approach which takes away the user from the necessity to care about an efficient resolving of imbalance of computations being performed in different processes and on different scales of his application. The efficient balance of internal workload in distributed and multi-scale applications may be achieved by introducing: a special multi-level model; a contract (or domain-specific language) to formulate the application in terms of this model; and a scheduler which operates on top of that model. The multi-level model consists of computing routines, computational resources and executed processes, determines a mapping between them and serves as a mean to evaluate the resulting performance of the whole application and its individual parts. The contract corresponds to unification interface of application integration in the proposed framework while the scheduling algorithm optimizes the execution process taking into consideration the main computational environment aspects.

Keywords: Multi-scale applications · Distributed computing · HPC Optimization · Multi-agent modeling · MPI

© Springer Nature Switzerland AG 2019

1 Introduction

The growth in the performance of computing systems (CS) for scientific computing and the increasing complexity of computer simulation models is one of the leading trends in the development of information technologies. Currently, the implementation of the exascale computing by 2020 is being discussed. On the other hand, this performance increase is mainly associated with the complexity of the CS architecture. Efficient use of this type of integrated CS in modelling is a complex engineering task. In addition to that, following challenges emerge:

1. the need to use multi-scale and multi-physical models, various modelling methods (grid and drains) in the solution of one applied problem;
2. the use of specialized computation resources (for example, graphics processors);
3. the problem of balanced spatial decomposition due to the complexity of the geometry of the domain of definition;
4. dynamic change in the complexity of different parts of the problem: with spatial decomposition due to the change in the geometry of the system or due to the emergence of areas of high computational complexity (for example, clustering of agents in multi-agent systems, slow convergence regions for grid methods;
5. the development of cloud computing technologies, in which the CS architecture is hidden from the user, and the need to meet their requirements.

This leads to a significant slowdown in the development of new simulation models, the use of obsolete technologies for parallel problem solving, the low efficiency of resource allocation and, ultimately, the inability to master the exascale computing system. In these circumstances, the traditional approach in which the responsibility for the parallel performance of software implementation of the model is assigned to the developer of the model is not efficient. Within the framework of this project, the approach to the separation of the development process of a simulation model from solving the problem of the efficient use of computing resources is given, which should be solved automatically. At the same time, efficient allocation of resources is impossible without knowledge of the internal logic of models; therefore, a tool should be proposed for its formal description in the resource allocation system and for providing the system with access to the task decomposition; dynamic allocation of resources should be ensured. The development of this approach for automatic resource allocation will significantly reduce the complexity of implementing computationally complex simulation models, allowing the developer to concentrate entirely on modelling methods, increase the efficiency of using computing resources. The authors of the proposal are sure that without the solution of these problems and the development of the proposed approach, the CS's exascale performance will not be accessible from a practical point of view. Additionally, simplifying the implementation of models stimulates interest in more complex modelling methods, for example, using multiscale approaches. In this paper, we present an approach for efficient execution of multi-scale distributed applications with the dynamic overflowing workload.

This approach includes dynamic (variable) graph model for allocating the structure of a multiscale distributed application as well as the unified framework for constructing this graph model and applying the algorithm of efficient management of executed tasks on dedicated computational resources.

2 Related Works

Since multiscale applications have a graph structure, the task of their planning is generally considered as the task of planning composite applications. To date, there is a wide variety of algorithms and methods for solving this problem. In [1], the authors proposed a coevolutionary genetic algorithm (CGA) for planning scientific composite applications that have execution deadlines. Experimental studies have shown the high efficiency of the developed method for optimizing the value of the resources used. However, this algorithm does not allow to optimize further the execution time, which reduces the possibility of its use. The heuristic IPEFT algorithm was proposed in [2]. The results of the experiments showed a fairly low execution time for small applications, as well as better results compared to the predecessors - HEFT and PEFT. Despite this, the authors' experience with heuristic algorithms [3,4] shows that their ability to find optimal solutions is very limited. An additional direction of research in the field of planning composite applications is the study of the ways to ensure energy efficiency of tasks. So in [5], the authors presented the heuristic EONS algorithm for planning composite computations taking into account the energy consumption of computing resources. But, since in most modern projects energy efficiency is not a key factor for optimization, it must be considered in conjunction with the implementation time and the cost of using resources. In [6] the methods of planning composite applications in the conditions of time constraints and the budget for the computing resources rent in the cloud environment are studied. The ideas of the planning algorithms proposed in the article are based on concrete, well-structured templates of the composite application. This approach is not always efficient because there are strict requirements for the structure of the composite application, which in general will rarely be met. There are works devoted to the development of systems for organizing the implementation and design of composite applications. For example, [7] presents a system for modelling designing and integrating composite applications into a computing environment. Such systems are aimed at simplifying the process of creating and executing applications. [8] presents a platform for organizing the planning process based on the flow of tasks and the dependencies between them. The main idea of the architecture of the platform is to present all the computational tasks in one composite application, which expands due to the tasks entering the platform. In [9], a detailed analysis of the types of multiscale applications, as well as possible ways of their implementation in a distributed environment, is given. The authors identified three types of applications: related, scalable and prioritized applications. For each type, application examples were selected and manual optimization was performed. As shown by the results of experimental studies, the use of knowledge about the nature of applications can significantly accelerate their implementation. However, the authors did not offer automated optimization paths, which is a critical drawback of the work. The cloud platform for analyzing and visualizing multiscale data is presented in [10]. The platform is based on the integration of tools and services for data analysis with services for data storage and composite applications execution. The main objectives of the platform development were to provide a convenient tool for modelling the processing of multiscale data, their implementation with automatic scaling of the computing environment and visualization of the analysis results (for example, climate data). In [11], the authors analyze the capabilities of existing composite application management platforms for efficient work with extreme-scale composite applications. Under extreme-scale composite applications, the authors mean applications that require advanced high-performance computing technologies for highly accurate predictive models based on the analysis of large volumes of multiscale data. The authors of [12] presented a modified version of the framework for distributed execution of multi-scale MUSCLE-HPC applications. Its main advantage over the previous version of MUSCLE-2 is the more efficient distribution of tasks in high-performance clusters by analyzing the relationships between applications and the location of closely related tasks within a single cluster. Despite the presented advantages, in MUSCLE-HPC, as well as in previous versions, there are no mechanisms for optimizing the applications themselves during the execution.

3 Multilevel Approach

This section describes basic parts of architecture of the proposed approach taking into consideration problem statement aspects.

3.1 Problem Statement

As mentioned before, the main problem lays between complexity of model distributed blocks interconnection and infrastructure appropriate mapping. Consider that application has execution environment that is organized as a computational grid $G\!<\!V,E\!>$, where $V=\{v_j\}$ corresponds to vertexes and $E=\{e_{j1,j2}\}$ represents edges. Upon the environment, grid computational elements $W=\{w_l\}$ form actual load of model logic. This elements may move in the environment from one nodes to another each computational iteration. V nodes are divided between computational resources $R=\{r_m\}$. Let consider $S=\{w_l^j\}$ as current distribution of actual load and define reorganization function

$$f(S_1,S_2)=\Sigma\frac{c_{w_l^j}}{e_{j,j'}}\cdot\delta_{j'}^{j}$$

$$\delta_{j'}^{j}=\begin{cases}1, & \text{if } j \text{ and } j' \text{ are on different resources}\\ 0, & \text{otherwise;}\end{cases}\quad,\forall j,j'=1,\ldots,J_m,$$

where $c_{w_l^j}$ - is amount of metadata needed to transfer load from v_j to $v_{j'}$; while $e_{j1,j2}$ corresponds to network channel throughput.

$$T(S) = max_m(\Sigma_j^{J_m} \frac{w}{p_m} + \Sigma_j \Sigma_{j'} \frac{w_j}{e_{j,j'}} \cdot \tau_{j'}^j)$$

$$\tau_{j'}^j = \begin{cases} 1, & \text{if there is actual load moving from } j \text{ to } j' \\ 0, & \text{otherwise;} \end{cases} , \forall j, j' = 1, \ldots, J_m,$$

Having these equations we can define the change environment criteria for the optimization algorithm:

$$T(S_{prev}) \cdot \theta > f(S_{prev}, S_{new}) + T(S_{new}) \cdot \theta.$$

Here θ is statistically depending value that corresponds to the rate of changing of actual load through the elements of the environment.

3.2 The Approach

Our approach of efficient execution of distributed applications consists of 3 main parts: a three-layer model of performance; a scheduling algorithm that uses the model to estimate performance of an application in different configurations; the partition-based model of computations that allows user to provide its own routines for computations. The basic concept is presented in Fig. 1.

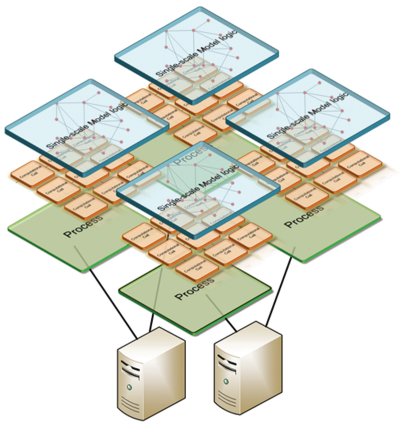

Fig. 1. Three-tier design of architecture

To solve the problem of scaling using the proposed model we developed a genetic-based algorithm to balance the workload on individual processes and thus improve overall performance. The objective of the algorithm is to reconfigure the computing resources based on the current load profiles generated by the computing processes on the resources. Concerning the task of modelling the behaviour of the population in specified urbanized areas (see City-Simulator application in Case Study section), the developed algorithm adaptively manages the allocation of modelling areas to physical computing processes based on the current specific load of these processes.

At the core of the model, there are few logical entities - “process”, “agent”, as well as the matrix of process contiguity with each other. Based on the fact that individual processes are responsible for modelling individual geographic areas, and moving agents across the city imply moving only between adjacent areas (in the simplest case), the adjacency matrix defines the subsequent area on the agent path at each time point. In this case, if modelling processes on nodes of computing clusters are placed arbitrarily, the absence of excessive network interaction is not guaranteed.

3.3 The Scheduling Algorithm

To perform scheduling and rescheduling of partitions among all processes in the distributed application, we implemented a special version of the genetic algorithm (GA) as a part of our approach. The genetic algorithm was chosen because its generality and ability to search through the whole solution space.

Our version of GA performs a search of optimal mapping between partitions and processes.

The mutation operator is implemented as random choosing of a host process for a random partition. As the crossover operator, the single-point crossover was chosen. To speed up convergence of the algorithm, the mutation may happen more than once per instance of the chromosome. The parameters of this GA stayed the same for all experimental runs and were the following: size of population - 100, count of generations - 300, mutation probability - 0.7, crossover probability - 0.3, selection operator - roulette wheel. The three-layer model was used as a fitness function to estimate the resulting execution time of modelling per iteration. The scheduling algorithm is used as before the start of the execution as during the runtime. In the latter case, the algorithm is being periodically run according to a shift between the last estimated execution time of iteration and the current value of that time.

3.4 The Partition-Based Model of Computations

To make the proposed approach working, it is necessary to introduce a model to describe required computations. This model is responsible for: (a) integrating of user-written computing functions into the framework based on the proposed approach and (b) obtaining required for the three-layer model monitoring and profiling data.

To achieve the stated goals, we propose the following model that can be easily expressed on any high-level programming language.

Let introduce main entities: $a_t^i = <s_t, x_t>$, $p_t^k = <S_p, \{a_t^i\}>$, $e_{t+1}^r = <\{a_{t+1}^j\}, l_{t+1}^a>$, where a_t^i is an agent, p_t^k is a partition, e_t^r is an envelope, all of them in moment t. An agent represents data the computing should happen on (x_t - location of the agent in modeling space, s_t - the rest of data associated with the agent). A partition represents an area in modeling space which has a set of associated agents with it and some static information required for the execution S_p. A partition serves as a unit of scaling. An envelope is a unit of data transferring between two connected partitions.

The user has to supply two functions m_u and g_u to be used for computing new state across all partitions and all agents on each node according to the following transformations.

$$p^k_{t+1} = m_u(p^k_t, \{e^r_t\})$$

$$l^a_{t+1} = g_u(x_{t+1})$$

$$<p^k_{t+1}, \{e^r_{t+1}\}> = f(m, g, p^k_t, \{e^r_t\})$$

where m_u - compute functions that implement the logic of modelling on the set of agents belonging to the area and multiple agent inflows, g_u - a function that determines a partition the agent should reside to according to its new coordinates x_{t+1} in the modelling space. Function f uses these user-defined functions, partitions data and incoming flows of the agent to calculate new states and perform all service functionality including optimization of data exchange between individual processes of the distributed applications.

4 Case Study: A Large Scale Multiagent Simulation of Urban Traffic

4.1 Urban Traffic Simulation

To carry out experimental studies of the performance and scalability of the proposed approach, a test case was developed for the field of multiscale modelling of urban mobility of the population in urbanized areas. It was called City-Simulator. In the developed example, we simulate the daily dynamics of people moving around the city, taking into account the specifics of these movements - from the sleeping areas to the city centre in the morning and from the centre to the sleeping areas in the evening. The city is divided into areas of modelling, the number of areas corresponds to the number of allocated computing resources allocated to the application. Each area and agents inside it are modelled in interrelation with other areas because, during the simulation, agents move between regions. This application assumes the dynamism of the computational load on the processes associated with the movement of agents by location, each of which is processed at the designated node for it.

Data on the mobility of the population on a city scale are simulated through the software package sim_city_package, after which the output array is transferred to the sim_district_package software package, responsible for multiagent modeling of individuals' behavior on the scales of individual regions (in units), the interaction between which does not involve large transmitted data and, as a consequence, is not a determining factor when planning placement on the nodes. The microscale modelling of the behaviour of agents within a set of small areas of the district (counted in hundreds, thousands and tens of thousands) is carried out by multiple copies of the package sim_object_package. The amount of communication between the instances of this package is most significant across the entire application, as a result of which it is the determining one when planning the placement of processes on the nodes, as provided by the extreme scaling (ES) pattern [13]. An example of dividing a city into regions (the city is divided into eight regions) and the general scheme of interaction taking into account intra- and inter-district communication are presented on Fig. 2).

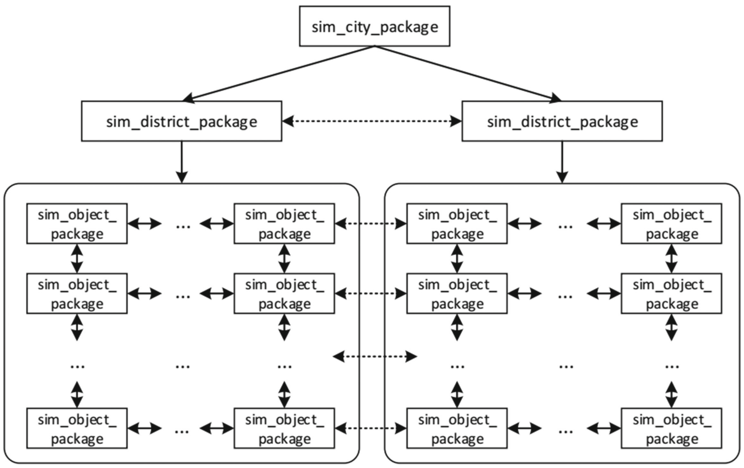

Fig. 2. Example of computations structure in distributed application City-Simulator

To configure the parametric performance models, the City-Simulator application is profiled depending on the size of the simulated area and the number of simulated agents. The experiments were conducted on the resources of the computing cluster of the University ITMO. The number of simulated agents ranged from 100 to 500 thousand in increments of 100 thousand. The width of the modelling area varied from 5 to 15 km in increments of 2.5 km. The dependence of the total calculation time and the time of data transfer between the modelling blocks on the number of processes on which the sim_object_package packet is placed, on one iteration is shown Fig. 3.

Figure 3 shows that the most intensive decrease in computation time is observed in the range of 4 to 32 processes. Also, there is a significant increase in the time spent on communication overhead, as the number of computing processes increases. The initial reduction in communication time in each of the scenarios (2–8 processes) is due to the placement of processes on one node from the calculation of 8 processes to one physical computing node, which does not cause network interaction between processes.

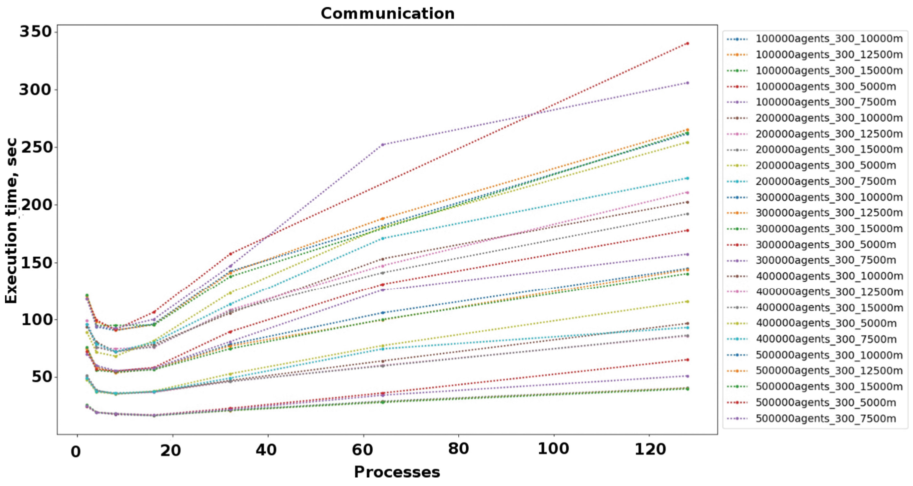

Fig. 3. City-Simulator execution time dependence on number of processes

The obtained results of performance profiling clearly show that the distributed application build with static partitioning of modelled areas doesn't scale well and eventually can't fully utilize available resources. With the growth of the simulation scale and preserving the same or close patterns of agents dynamic - e.g. increases in the number of agents and/or a number of areas - the problem with efficient resources utilization is getting worse. The latter makes the user wait more and thus slows down research speed. Using the proposed approach is possible to ease the pain caused by this problem by improving scaling capabilities of the application exploiting patterns in its dynamic.

For this test case, the application execution scheduling component can provide planning optimization through the methods described in The approach section - accounting for agent movements between modelling areas when scheduling tasks and load balancing of compute nodes by reconfiguring the grid.

4.2 Experimental Results

The experiments were carried out - with and without application of adaptations of the computational template to the application. Activity modelling was carried out for 6 million agents to produce simulations on a city scale both regarding the size of the calculation area and regarding the size of the population of agents. The application was planned for 100, 200, 500 and 1000 cores. The results of the experiments are shown in Fig. 4. As can be seen from the graph, for experiments without the use of adaptations, the total simulation time with changing in the number of processes from 100 to 200 decreases (by 9%), but with a further increase in the number of computing cores, the total execution time increases (by 7% at 500 cores and 33% for 1000 cores). Such an effect, first of all, is due to the fact that when planning tasks, the adjacency matrices of the movements of agents are not taken into account, which entails a considerable increase in the network interaction between the processes. In addition, the lack of reconfiguration of the calculated grid also slows down the execution of the application due to the uneven load on modelling processes. For the experiment with the use of adaptation, we can see a decrease in execution time by 32% in the first case and by 18% in the second one. Despite the slowing down of the rate for execution speedup, the proposed approach can deliver significant improvement in efficiency of resource utilization and thus to scale for the distributed application.

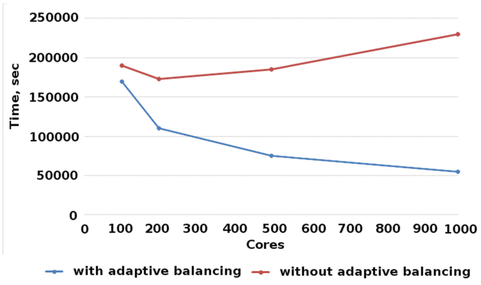

Fig. 4. Performance comparison of classical static partitioning approach and the proposed approach in case of City-Simulator distributed application

The results of the experiments in the scenario using the modelling template adaptations demonstrate a stable decrease in execution time with an increase in the number of computational cores. However, it is worth noting that the change in execution time slows down as the number of cores increases. This is due to the fact that even with the use of adaptations of the computational template with a large number of modelling processes, the contribution of network interaction inevitably increases, which is not capable of completely levelling even the InfiniBand data transmission channel.

To further analyze the results obtained, changes in the computational load of modelling processes over time were investigated. Since the computational load is expressed in terms of the number of simulated agents in a certain process, for the simulation experiment with 1000 computational cores, data was collected on the number of agents at each time point and the dynamics of the change in the number of agents for scenarios with and without reconfiguration of the grid was analyzed. Figures 5 and 6 show the graphs of three processes in which the characteristic effects of the optimization performed by the PC EO are visible

For the process, the graphs of which are represented in Fig. 5, by reducing the cell size it is possible to reduce the computational load on average from 2500 to 1000 agents per iteration. In the process, the graphs of which are shown in Fig. 6, the situation was the reverse - at a certain stage of modeling the load dropped and fell almost to zero. By distributing the load from other processes it was possible to increase the workload of the process, thereby reducing the probability of downtime.

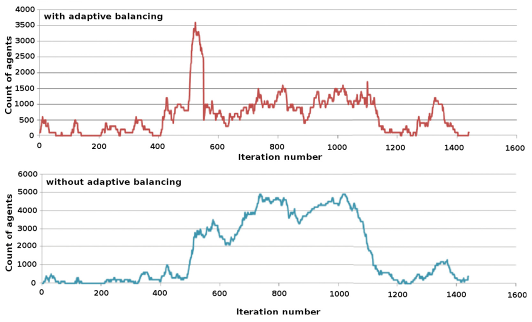

Fig. 5. Workload decreasing for a processes in City-Simulator

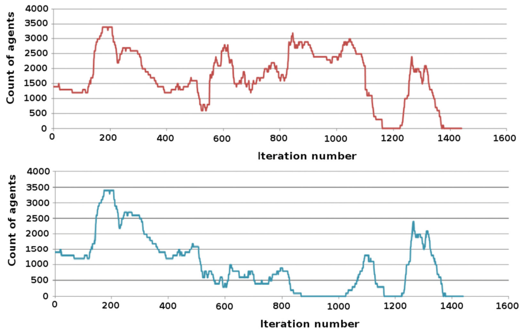

Fig. 6. Workload increasing for a processes in City-Simulator

Analysis of simulation results in individual processes showed that the reconfiguration of the computational grid allows relatively equalizing the computational load on the processes that perform agent modeling of the movement of people in the urban environment. This factor, coupled with the contiguity matrices in the planning process, allows you to significantly optimize the execution of the application and ensure its scalability on a large number of compute nodes.

5 Conclusion

The experimental case study demonstrated an improvement in execution time with the growth of exploited cores for: up to 32% in case of 200 cores, up to 18% in case of 500 cores thus showing that the proposed approach is able to deliver significant improvement in efficiency of scaling for distributed applications. This research is financially supported by The Russian Science Foundation, Agreement #14-11-00823.

References

1. Liu, L., Zhang, M., Buyya, R., Fan, Q.: Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. Concurr. Comput. Pract. Exp. **29**(5) (2017)
2. Zhou, N., Qi, D., Wang, X., Zheng, Z., Lin, W.: A list scheduling algorithm for heterogeneous systems based on a critical node cost table and pessimistic cost table. Concurr. Comput. Pract. Exp. **29**(5) (2017)
3. Visheratin, A.A., Melnik, M., Nasonov, D.: Dynamic resources configuration for coevolutionary scheduling of scientific workflows in cloud environment. In: Pérez García, H., Alfonso-Cendón, J., Sánchez González, L., Quintián, H., Corchado, E. (eds.) SOCO/CISIS/ICEUTE -2017. AISC, vol. 649, pp. 13–23. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67180-2_2
4. Visheratin, A.A., Melnik, M., Nasonov, D.: Automatic workflow scheduling tuning for distributed processing systems. Procedia Comput. Sci. **101**, 388–397 (2016)
5. Chen, H., Zhu, X., Qiu, D., Guo, H., Yang, L.T., Lu, P.: EONS: minimizing energy consumption for executing real-time workflows in virtualized cloud data centers. In: 2016 45th International Conference on Parallel Processing Workshops (ICPPW), pp. 385–392. IEEE, August 2016
6. Wang, Y., Shi, W., Kent, K.B.: On optimal scheduling algorithms for well-structured workflows in the cloud with budget and deadline constraints. Parallel Proc. Lett. **26**(02) (2016). https://doi.org/10.1142/S0129626416500092
7. Balis, B.: HyperFlow: a model of computation, programming approach and enactment engine for complex distributed workflows. Future Gener. Comput. Syst. **55**, 147–162 (2016)
8. Zenmyo, T., Iijima, S., Fukuda, I.: Managing a complicated workflow based on dataflow-based workflow scheduler. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 1658–1663. IEEE, December 2016
9. Borgdorff, J., et al.: Performance of distributed multiscale simulations. Phil. Trans. R. Soc. A **372**(2021) (2014). https://doi.org/10.1098/rsta.2013.0407
10. Lu, S., et al.: A framework for cloud-based large-scale data analytics and visualization: case study on multiscale climate data. In: 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), pp. 618–622. IEEE, November 2011
11. da Silva, R.F., Filgueira, R., Pietri, I., Jiang, M., Sakellariou, R., Deelman, E.: A characterization of workflow management systems for extreme-scale applications. Future Gener. Comput. Syst. **75**, 228–238 (2017)
12. Belgacem, M.B., Chopard, B.: MUSCLE-HPC: a new high performance API to couple multiscale parallel applications. Future Gener. Comput. Syst. **67**, 72–82 (2017)
13. Alowayyed, S., Groen, D., Coveney, P.V., Hoekstra, A.G.: Multiscale computing in the exascale era. J. Comput. Sci. **22**, 15–25 (2017)