Development of simulation model of HPC system for Super Charm-Tau factory

D Weins¹, V Vorobyev², I Chernykh¹ and I Logashenko²

¹ The Institute of Computational Mathematics and Mathematical Geophysics SB RAS (ICM&MG SB RAS) 6, Acad. Lavrentieva Pr., Novosibirsk, 630090, Russian Federation

² Budker Institute of Nuclear Physics of Siberian Branch Russian Academy of Sciences (BINP SB RAS) 11, Acad. Lavrentieva Pr., Novosibirsk, 630090 Russian Federation

vins@sscc.ru, V.S.Vorobev@inp.nsk.su, chernykh@ssd.ssc.ru, ivan.logashenko@gmail.com

Abstract. The article describes the design of a digital model of an HPC system for processing data from the Super Charm-Tau factory electron-positron collider of the "megascience" class. This model is developed using the AGNES multiagent modeling platform. The model includes intelligent agents that mimic the behavior of the main subsystems of the supercomputer, such as a task scheduler, computing clusters, data storage system, etc. This model allows calculating the parameters of the computing system necessary for processing and storing the results of operation of the Super Charm-Tau factory after its commissioning.

1. Introduction
The «Super Charm-Tau factory» (SCTF) project, which is a symmetric electron-positron collider of ultrahigh luminosity with a beam energy at the mass center system from 2 to 6 GeV [1], is developed at the Budker Institute of Nuclear Physics, Siberian Branch, Russian Academy of Sciences (BINP SB RAS). This project comprises a unique accelerating-storage complex with a luminosity of \(10^{35} \text{ cm}^{-2}\text{s}^{-1}\) and a universal elementary particle detector. The main goal of experiments carried out on the SCTF is to study the properties of tau lepton and charmed particles, subject the existing microworld theory and Standard Model to high-presicion verification, and to search for phenomena not described within the framework of this theory.

In the course of the experiments, about 100 petabytes of “RAW” data is accumulated from the elementary particle detector of the SCTF. An important role in the project is played by the system for data processing and storage, whose tasks include the primary data processing, data transfer to long-term data storage system (decades), data extraction from the storage system for processing and processing using high-tech computing (HPC) systems. Specialized software should allow one to analyze the accumulated data by a collective of about 1000 physicists. The development of the data analysis algorithms and the optimization of the detector structure are carried out using modeling data generated via software for modeling of experiments.

The vast volumes of primary data require large investments into the creation of the localized computational infrastructure of the complex. Within the framework of this paper, it is proposed to
solve the problem of designing this computational infrastructure and multilevel systems for storing the experimental data using simulation modeling. This approach allows for preliminary estimation of the structure and amount of data generated by the physical device and determination of the parameters and configuration of computational infrastructure required to solve the experimental problems.

2. Selection of the simulation modeling system

2.1. Multiagent modeling systems
Agent modeling (AM) is an advanced approach to modeling systems containing autonomous and interacting intelligent agents. An intelligent agent is an entity existing in a medium and possessing sensors for perceiving this medium and executive mechanisms for affecting this medium. The agents operate asynchronously according to their laws and interact with other agents in order to reach common goals. During operation, a software agent can change both the outer medium and its own behavior [2]. A distributed, asynchronous behavior is quite important for constructing a simulation model of a supercomputer as it is virtually impossible to ensure the centralized control of tens and hundreds of millions of cores.

A multiagent system (MAS) is a system formed by several interacting intelligent agents in a single medium. Due to such properties as autonomy, limited knowledge, decentralization, and self-organized and intelligent behavior, the agents naturally fit the modeling of inhomogeneous systems with atomic elements, which also include HPC systems [3].

2.2. AGNES
The AGNES system, initially developed for telecommunication and information networks [4], also demonstrated its efficiency in modeling the operation of a distributed control system and executing high-performance parallel programs [5].

The AGNES package is based on Java Agent Development Framework (JADE), serving as a means for creating Java based MAS [3]. JADE is a powerful tool for developing Java based multiagent systems. It is important for modeling large computations that JADE is a FIPA-compatible, distributed agent platform, which can use one or several computers (network nodes), each having only one operating JAVA machine.

All interaction between JADE agents occurs by exchanging messages according to FIPA specification. The key property of a JADE agent is a set of its «behaviors». An agent's lifecycle ends when this agent has no active behaviors. AGNES uses the advantages provided by JADE and expands the multiagent system to a modeling system. AGNES comprises two types of agents: functional and controlling agents. The primer perform simulation, and the latter control and ensure operation of the primer.

In order to ensure high-level fault tolerance, AGNES implements the basic principles of improved fault tolerance of the modeling medium: storage decentralization and information redundancy.

AGNES is a distributed MAS referred to as a platform. AGNES consists of a system of containers distributed over a network. As a rule, each host has one container. Agents exist within containers. As agents differ in their functions and tasks, AGNES attempts to uniformly redistribute the agents of all types.

Thus, AGNES is a perfect fit for the development and control of the operation model of a HPC system.

3. Description of the model

3.1. General scheme of the HPC system model
The model of the HPC system for data storage and processing with SCTF is developed under the following assumptions:
The HPC system under consideration is a distributed, heterogeneous, and multicluster computing system (CS) with an MPP architecture. The task flow fed to the HPC is distributed over the clusters according to a certain algorithm. The tasks are implemented without any interaction among the clusters. The HPC system contains a parallel data storage system, with all of the system clusters having access thereto.

A cluster is regarded as a set of single-type elementary machines (EMs) related to each other by some communication scheme, in which EMs are nodes and the communication lines between them are branches.

The main resource given to solve the problem is a virtual domain of computational nodes (CNs), satisfying all the stated parameters of the problem. A CN corresponds to an EM in a real system.

If the main characteristic of solving the problem for a real CS is time, then the number of elementary operations executed by a task is taken a main characteristic in the model.

Transition from model time to real time is carried out using a special coefficient.

The general scheme of the multiagent model of the HPC system, which accounts for the above-given assumptions, is illustrated in figure 1.

All types of objects given in the scheme are intelligent agents possessing their own set of parameters and behavior algorithms.

3.2. Types and functions of agents in the model of the prompt management system of the HPC system
Several types of agents can be seen in the scheme in Fig. 1:
- users at random instances send their parallel tasks to the HPC system;
• the detector with a certain frequency transfers data of certain size to the data storage system of the HPC system;
• the system for data collection and statistical analysis gathers and stores generalized information about tasks solved using the HPC system;
• the distributor distributes the flow of incoming tasks among the clusters included in its composition with account for their state at a current time;
• the collector of the CS schedules the execution of incoming tasks and arranges the implementation of this schedule;
• the data collector ensures that data are saved and read from storage devices of the HPC system;
• the controller of the CS domain arranges correct operation of the CSs under its control;
• the CS simulates the actions of the EMs, executing its branch of the parallel program or being in a stand-by mode.

The multiagent model of the HPC system accounts for combining the agents that perform at a current instance one and the same function into virtual domains (below simply referred to as domains), controlled by the controller agent of the CS regions. Thus, the CS domain is combining the group of agents of computational nodes and the controller agent of the domain, executing at a current instance one and the same function and connected by logical communication lines. During the operation of the HPC system model, the following domains are formed:

• the CN reserve domain contains computational nodes sent into “cold” or “hot” reserve [6], as well as nodes decommissioned for maintenance or repair.
• the domain of the free CNs contains all the CNs of the cluster, which are neither preoccupied by executing a user’s task nor in reserve at a given instance;
• the domain of operating CNs contains computational nodes preoccupied by solving one and the same task at a given instance, in which case the controller agent observes both the physical characteristics of the nodes and the course of execution of the parallel program.

3.3. Arranging the execution of parallel tasks in the HPC system model

The user’s tasks are fed to the distributor input of the high-tech CS. The distributor of the HPC system constantly receives reports from the collector agents about changes in the state of their task queue and available resources. Initially, the switch connects the agent of collection and analysis of statistics. The latter looks through this user’s tasks and selects those similar in requested resources. The commuting switch is given a weighted list of preferences of the CS collector agents. Guided by the data received from the agent of collection and analysis of statistics and the available data on the workload of the collector agents, the switch sends the task to the most promising one (the least loaded one) having the largest weigh in the list of preferences.

This, the task is fed to the corresponding collector agent of the CS, and the currently running task schedule is immediately reviewed. If resources are found in this schedule for the task, it is embedded into the schedule, otherwise it is sent to the task queue. As soon as the current task is executed, the collector agent develops a new schedule. Tasks are selected from the queue using one of the planning algorithms.

The task is being executed as soon as its turn arrives. For this purpose, a domain controlling agent is created (or chosen from those already formed). This agent takes control of the CNs from the domain of free CNs. Nodes for the created domain should be maximally similar in the graph of the communication medium of the CS. The controller sends a request to the data controller for a time required to read a certain volume of information for the problem solved and simulates the reading process from the data storage system into the local memory. Then, as the controller agent of the
operating domain periodically collects information about the course of the task execution, it waits for the execution to end.

As the task is executed, its completion parameters are sent back on command until reaching the agent of collection and analysis of statistics, where the final stage of the task execution occurs (its execution statistics is saved). This statistic can be used to estimate the amount and characteristics of computational resources required for the computing system to execute a desired number of user’s tasks without serious delays.

3.4. Fault tolerance and energy efficiency
The dynamic fault tolerance system is implemented in this model using the CN agents and domain controlling agents.

The CN agent collects information about the temperature, load, and condition of the communication lines between the nodes and other information that characterizes the CN state. Each CN agent has a set of behavior rules for preventing the CN failures. As soon as one of the parameters is out of range or the trend of changes of this parameter becomes worse (for example, the temperature rapidly rises), the CN agent sends a signal to the controller agent of its domain.

The domain controlling agent receives information about an emergency situation occurring in any node and commands all nodes to stop operating. Next, the received signal is analyzed. If the nodes report loop and deadlock, then the task execution is canceled. If one of the nodes signals violation of its physical parameters (temperature rise or communication line disconnection), the controller agent searches for a node to which the computational load is transferred from the faulty node. Initially, the resources located in a given domain are analyzed. If they are insufficient, the controller connects with the controllers of the domain of free nodes, reserve domain, and “neighboring” operating domains. Next, the computational load is transferred to a faultless node, while the faulty node is transferred to the reserve domain. If the failure cannot be prevented and the node fails during task execution, the
domain controlling agent searches for a replacement and then starts the task again from the last execution point.

When it comes to the energy efficiency of cluster computing systems, the following problems may be encountered:

- There are idle cluster nodes that consume significant amounts of electricity;
- Nonuniform temperature distribution over the cluster nodes with a large amplitude, which reduces the reliability and fault tolerance of the cluster and leads to an ineffective use of electricity.

The primer problem is solved in the model using the reserve domain. This domain can be divided into three regions: hot reserve, cold reserve, and nodes requiring repair or maintenance. As the model is launched into the reserve domain (hot reserve), at least 5% of the CNs of the CS are determined. These CNs can be used for the dynamic balancing of the load by the domain controllers of the operating nodes. As soon as the node authorized to perform any task becomes free, it returns to the reserve domain. If the CS collector analyzes the formed schedule and task queue and concludes that there is a certain number of nodes that will remain idle in the foreseeable future, it commands to send them to the reserve domain.

The latter problem is solved by the collector agent of the CS as it forms another schedule. The presence of emergency shutdown thresholds upon reaching a critical temperature is not enough for efficient energy planning [7]. Because of this, the CS collector gathers information about the system nodes and analyzes the temperature trend for the last second. Depending on the fact whether the node processor temperature drops or rises, the trend approaches unity or, conversely, tends to zero. At the instance when the CS collector begins forming a new schedule, it places tasks in the nodes with the lowest value of the trend in a descending order. Thus, the temperature distributes over the cluster nodes more uniformly.

The use of energy efficiency and fault tolerance schemes in the data model allows estimating a required number of additional computing resources that ensure the smooth operation of the HPC system.

3.5. Parallel access to data

The tasks of storage and processing of the data sent from the SCTF in the HPC system model are executed by simulating the operation of the parallel data storage system. Figure 3 shows the scheme of the communication between the data collector agents, the domain controllers executing parallel tasks, and the agents simulating the data coming from the detector and detector event modeling systems. The detector itself and the modeling system are represented by black boxes.
Each data collector agent contains a counter of saved data and information about the state and characteristics of all shelves with data carriers under its control. As the controllers of operating domains send a request to data collector agent, it transfers information about the time required to read and save the desired amount of data.

As seen from the scheme shown in Fig. 3, the HPC system model contains six data controllers, each storing only certain information: the database with detector calibration information, raw data from the detector and from the event modeling system, data from reconstructed real and model events, and raw data backup from the detector. This division is necessary in the model in order determine how many data carriers are needed to ensure the smooth operation of the SCTF.

4. Conclusion
The successful launch of the SCTF requires estimating the computational infrastructure parameters of the complex for storing and processing data of the physical experiment at the stage of design. Using simulation modeling allows for the maximally reliable representation of the exact characteristics and volume of the needed equipment for developing the desired HPC system. The simulation model described in this paper accounts for all the aspects of operation of this system from parallel data storage system to arrangement of the parallel launch of tasks. The developed system for processing software errors and equipment failures, as well as the system for ensuring energy efficiency make it possible to estimate the needed equipment with account for all possible emergency situations.

Thus, the developed simulation model allows calculating the parameters of the computing system required for processing and storing the operation results of the Super Charm-Tau factory after its commissioning.

Acknowledgments
This paper was financially supported by the Russian Science Foundation (Grant No. 19-72-20114) using the resources of the Siberian Supercomputer Center cluster (ICM&MG SB RAS, Novosibirsk).

References
[1] Bondar A 2013 Project of a super charm-tau factory at the Budker Institute of Nuclear Physics in Novosibirsk Phys. At. Nucl. 76 pp 1072–1085.
[2] Chandy K M and Misra J 1978 Distributed Simulation: A Case Study in Design and Verification of Distributed Programs IEEE Transactions on Software Engineering SE 5 N5 pp 440-452.
[3] Wooldridge M 2002 Introduction to MultiAgent Systems (England: JOHN WILEY & SONS LTD) p 348.
[4] Bellifemine F L, Caire G and Greenwood D 2007 Developing Multi-Agent Systems with JADE (Wiley) p 300.
[5] Podkorytov D, Rodionov A and Choo H 2012 Agent-based simulation system AGNES for networks modeling: review and researching Proc. Int. Conf. on Ubiquitous Information Management and Communication (Kuala Lumpur, Malaysia) P115
[6] Podkorytov D, Rodionov A, Sokolova O and Yurgenson A 2010 Using Agent-Oriented Simulation System AGNES for Evaluation of Sensor Networks LNCS 6235 pp 247–250.
[7] Glinskiy B M, Kulikov I M, Chernykh I G, Snytnikov A V, Sapetina A F and Weins D V 2017 The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers CCIS 793 pp 278-289
[8] Glinskiy B, Sapetina A, Martynov V, Weins D and Chernykh I 2017 The Hybrid-Cluster Multi-level Approach to Solving the Elastic Wave Propagation Problem CCIS 753 pp 261-274.
[9] Glinsky B, Rodionov A, Marchenko M, Podkorytov D and Weins D 2012 Scaling the Distributed Stochastic Simulation to Exaflop Supercomputers Proc. Int. Conf. on High
Performance Computing and Communications (Liverpool, England, UK) pp 1131–1136.

[6] Kallis J M and Norris M D 1996 Effect of steady-state operating temperature on power cycling durability of electronic assemblies Tech. report Hughes Aircraft Company pp 1.

[7] Coskun A K, Rosing T S and Whisnant K 2007 Temperature Aware Task Scheduling in MPSoCs Proc. Int. Conf. on Design, Automation & Test in Europe Conference & Exhibition (Nice, France) pp 1659 – 1664.