CYBERNETICS AND PHYSICS, VOL. 8, NO. 4, 2019, 195–198

DEVELOPMENT OF THE DISTRIBUTED INFORMATION SYSTEM FOR THE COOPERATIVE WORK UNDER THE DESIGN AND OPTIMIZATION OF CHARGED PARTICLE ACCELERATORS

Vladislav Altsybeyev
Saint Petersburg University
Russia
v.altsybeev@spbu.ru

Vladimir Kozynchenko
Saint Petersburg University
Russia
v.kozynchenko@spbu.ru

Article history:
Received 18.09.2019, Accepted 05.12.2019

Abstract

A problem of the development of distributed information system for carrying out numerical experiments in the purpose of design and optimizations of charged particle accelerators is discussed. The infrastructure of a considered informational system should provide an effective methods for organizing calculations, storing, sharing, exchanging and analyzing the results of numerical experiments and data, remote user access to the system for group of researchers. To achieve this aim we proposed distributed architecture based on independent components: database server, high-performance distributed calculation services, web interface that provides tools for interaction of users with information system. To ensure flexible changing of models we proposed the ”lazy filling” conception of database table.

Key words

Accelerators, information system, database, distributed computations, optimization

1 Introduction

The problem of developing new accelerator systems and improving old ones is related to the accumulation of a large amount of information about various design options and the carrying out a large amount of numerical experiments to determine their quality against the specified criteria. Storing the information and conducting experiments on the personal computing resources of the participants of the research team seems to be inefficient due to the data decentralization and the long duration of the calculations. So, development of distributed information tools, which allow efficient processing of the large amounts of data with the possibility of experiments data sharing based on providing centralized remote access for a group of researchers and engineers involved in the design process is a fundamental problem.

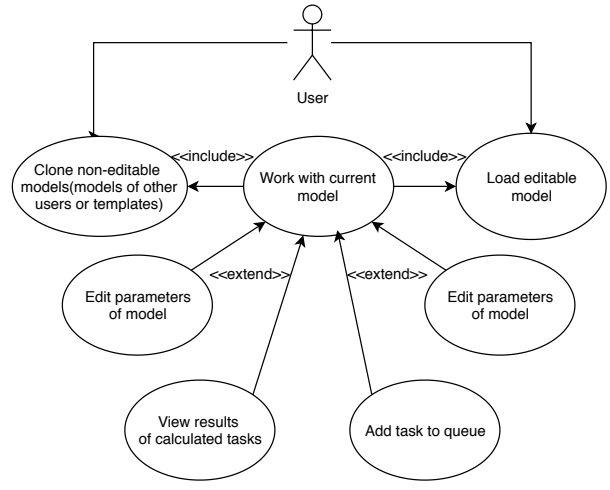

Figure 1. UML use cases diagram

The most appropriate way to solve this problem is to create an infrastructure consisting of a dedicated database server containing information on the parameters of accelerating structures, a queue of tasks to be performed, a history of the user actions, already calculated results, and a high-performance distributed calculation services for tasks from the queue. It is reasonable to provide remote access to the team members to the designated services using a web interface that provides tools for editing information from the database, queuing tasks and visualizing results. Thus, the combination of these

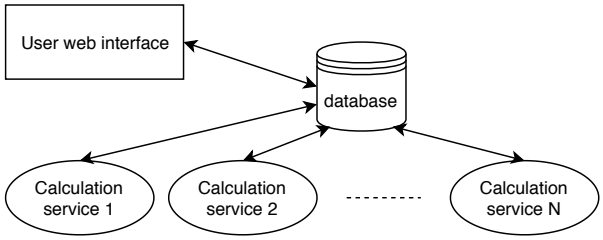

Figure 2. Main components of the information system

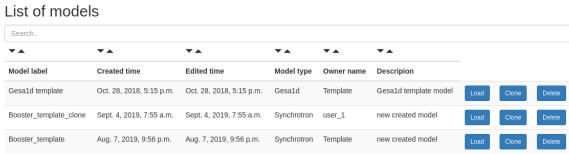

Figure 3. The main page

three components allow us to build a distributed high-performance information system that provides flexible, convenient and efficient tools for solving scientific problems. Using the resources of the Computing Center of St. Petersburg State University will allow us to organize a set of the dedicated VPS for the deployment of the proposed infrastructure with the secure network interaction and high performance, providing the adequate information processing times, centralized storage and a high performance of a distributed computing that are not available on the personal computers.

For example at present the main purposes of the discussed system is solving the problems related with the MegaScience NICA Accelerator Complex (Nuclotron-based Ion Collider fAcility) created at the Joint Institute for Nuclear Research (Dubna, Moscow region), aimed at solving the fundamental problems modern physics [Altsybeyev et al., 2018b; Altsybeyev et al., 2018c] and simulations of charged particles sources and linear accelerators [Ovsyannikov et al., 2014; Altsybeyev et al., 2018a; Altsybeyev, 2017].

2 Information System use Cases

Let us introduce the following definitions within the considered system:

The **Model** — the aggregation on data described the initial data for the experiments and links on the performed experiments results. There are different model types according to the solved problem types. For example, models for cycling accelerators are differ from models for the linear ones. Type of model determines the set of the initial parameters and list of different for each model type **solvers** (numerical algorithms with determined input and output). There are template private models which are available for editing only for administrator. Another models belongs to the

User — the account belongs to researcher. The user can edit his own models and view shared data belonged to the another users. Also it is possible to clone edit-lock models (template or another users models) with the aim to edit their without changing the original model. The user can run some solver in the model and it adds the **task** to the tasks queue.

Task is the entry consist of the aggregation of the reference to the corresponded **model**, reference to the solver should be executed, execution status and results data if solver is already executed. For executed tasks user can study results in a plotted or a text data.

The UML use cases diagram of the described information system is presented in Fig. 1.

3 Information System Description

3.1 General Relations

Main components of the information system and they interaction are presented in Fig. 2.

3.2 User Interface

The user interface is implemented using standard tools for the web application development. The main page (Fig. 3) contains full list of the model available in the database. The user can clone all and load his own models. The model page (Fig. 4) contains the lists of the editable parameters groups of the model and solvers(model components), queue of the tasks with general information(name, execution status, time of main events, etc) and the control buttons, two areas for data plot visualization, services monitor table and the area with logging data. All performed operations are logged to the database, these logs are available for reading in the user interface.

3.3 Database

The database provides all interaction between the user interface and the calculation services. The database relationships are presented in Fig. 5.

The table *owners* contains a description of the system users with names and id. The table *modeltypesnames* contains name of the types of the models currently available in the system. It is enough simple to add a new model type. To do it one have to add a new model type name to the table *modeltypesnames* and describe the model parameters joined in groups(model components) in the tables *modelcomponentstypesnames*, *viewstypesnames*, *modelcomponentrelations*, *modelcomponentsviewsrelation*, *modelcomponentsviewslabels*, *modelcomponentsviewssimpleitempattern*. All solvers is also model components. To add solver additionally one have add records to the table *plotdata*. Sometimes during development a new models we are facing with a such problem — if expanding models of a certain type is necessary(adding a new solver for example) it may be difficult to change all models of such type. We propose and implement the solution of this problem based on ”lazy filling” table rows there

Figure 4. The model page

concrete data of new model components are stored. This trick based on the following idea: the rows of the table *lazymodelcomponentsviewsitems* are filled only then we try to get data from this table. If the required record is missed, the required record will be created and inserted with some default content from the table *modelcomponentsviewssimpleitempattern*. Additional mechanic with the model components views allows us to represent the model component using different ways. So, after creation a new model a new record will be added only to the *models* table. After a new tasks adding by user a new record to the table *results* with state "In queue" (from table *resultstates*) will be added.

3.4 Calculation Services

Calculation service is implemented using C++. It is possible to run an arbitrary number of services that will run on the different servers with the unique database server. For solving a problems related with particle accelerator design and simulation we use MAD-X code [MAD, 2011]. Its scripting language is de facto the standard to describe particle accelerators, simulate beam dynamics and optimize beam optics at CERN. We build MAD-X on the Linux platform as shared object file and link with our calculation services. It allows us to do fast calls of MAD-X routines without a creation a new process as in a standard scenario of MAD-X using. Fast calls is especially important when we use our custom optimization methods which require significant number of calls MAD-X routines with the different inputs.

The high performance back-end services continuously browse the tasks queue. Then the new task is appeared, the service starts to execute it. Service reads corresponded data for the task from database, builds objects which represent model data and starts to execute corresponded routine. The result of solver routine execution is a structure with data described plots which should be visualized and text data. As a result, this structure is inserted in the database and then available for the visualization using user interface.

4 Current Status

At present model for studying beam dynamics in synchrotrons and transport channels is implemented and continue to be developed and tested. The following abilities are currently available:

1. calculation of nonlinear transversal beam dynamics based on MAD-X PTC module (example of calculation is presented if Fig. 4).;
2. optimization of magnets with different effective length arrangement;
3. Fourier analysis of errors of magnets and quadrupoles;
4. calculation of distribution of betatron frequencies and resonances;
5. correction of charged particle beam orbit in synchrotrons and transport channels (example of correction is presented if Fig. 6).

5 Conclusion

Infrastructure of the developed information system allow us to perform the cooperative work in the researchers team under the design and optimization of the charged particle accelerators. At present works on development of new algorithms and features for solving problems related with the MegaScience NICA Accelerator Complex in progress.

Acknowledgements

The authors wish to thank O.S. Kozlov, V.A. Mikhaylov, D.A. Ovsyannikov, A.O. Sidorin, A.V. Tuzikov, G.V. Trubnikov from Joint Institute for Nuclear Research for useful discussions on problems related to NICA Accelerator Complex and attention to the work done. This work was supported by Grants Council of the President of the Russian Federation.

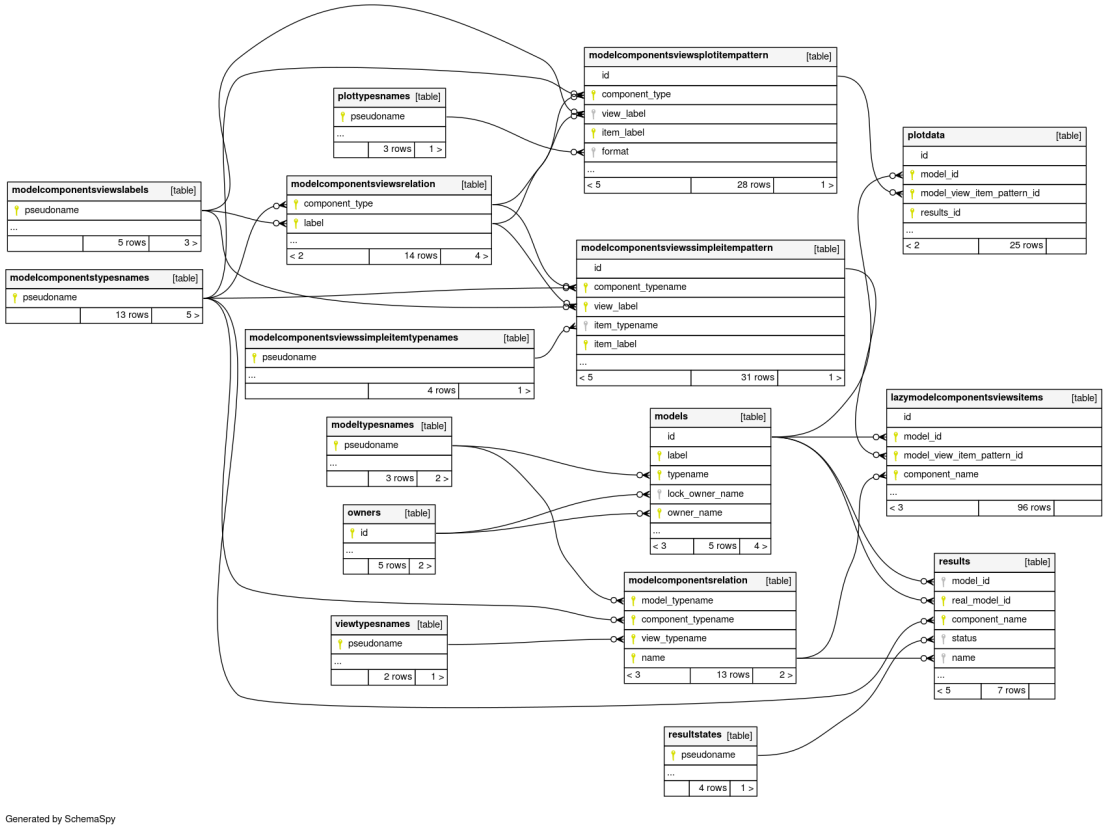

Figure 5. Database relationships

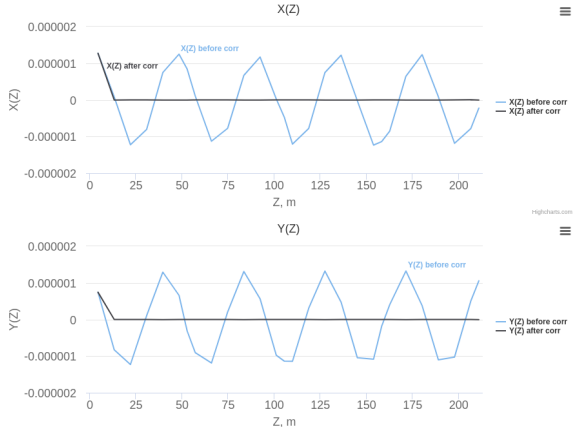

Figure 6. Example of the orbit correction in the booster

References

(2011). MAD — Methodical Accelerator Design. CERN-BE/ABP Accelerator Beam Physics Group. http://madx.web.cern.ch/madx/. Accessed: 2019-09-06.

Altsybeyev, V. (2017). Configurable code for beam dynamics simulations in electrostatic approximation. In *2016 Young Researchers in Vacuum Micro/Nano Electronics, VMNE-YR 2016 - Proceedings*, number 7880398.

Altsybeyev, V., Svistunov, Y., Durkin, A., and Ovsyannikov, D. (2018a). Preacceleration of the multicharged ions with the different A/Z ratios in single radio-frequency quadrupole(RFQ) channel. *Cybernetics and Physics*, **7**, pp. 49–56.

Altsybeyev, V. V., Butenko, A. V., Emelianenko, V., and et al. (2018b). Simulation of Closed Orbit Correction for the Nuclotron Booster. *Physics of Particles and Nuclei Letters*, **15**, pp. 854–857.

Altsybeyev, V. V., Kozlov, O., Kozynchenko, V., and et al. (2018c). Development of Software for Simulating and Analyzing the Dynamics of Charged-Particle Beams in Synchrotrons and Beam Lines. *Physics of Particles and Nuclei Letters*, **15**, pp. 798–801.

Ovsyannikov, A. D., Ovsyannikov, D. A., Altsybeyev, V. V., Durkin, A. P., and Papkovich, V. G. (2014). Application of optimization techniques for RFQ design. *Problems of Atomic Science and Technology*, **3** (91), pp. 116–119.