International Conference "Distributed Computing and Grid-Technologies in Science and Education"

June 30 – July 5, 2014, Dubna, Russia

GRID'2014

Book of Abstracts

JOINT INSTITUTE FOR NUCLEAR RESEARCH
LABORATORY OF INFORMATION TECHNOLOGIES

DISTRIBUTED COMPUTING AND GRID-TECHNOLOGIES IN SCIENCE AND EDUCATION

Book of Abstracts of the 6th International Conference

Dubna, June 30 – July 5, 2014

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ И ГРИД-ТЕХНОЛОГИИ В НАУКЕ И ОБРАЗОВАНИИ

Тезисы докладов 6-й Международной конференции

Дубна, 30 июня – 5 июля 2014 г.

Dubna 2014

Technologies, architectures, models, methods and experiences of building distributed computing systems. Consolidation and integration of distributed resources

REVIEW OF JULIA PROGRAMMING LANGUAGE FOR SCIENTIFIC COMPUTING

GEVORKYAN Migran, KULYABOV Dmitry, SEVASTYANOV Leonid
mngevorkyan@sci.pfu.edu.ru, yamadharma@gmail.com, leonid.sevast@gmail.com

Peoples' Friendship University of Russia

Julia is new high-level programming language that supports multiple programming styles and primarily focuses on scientific computing [1]. Language has been developed at the Massachusetts Institute of Technology (MIT) in 2009 under the MIT license and is available through the public repository on GitHub [2]. Language is under active development and it's latest version is 0.3.

Language has built-in tools for parallel and distributed computing, and supports multithreading (coroutines). Developers have mentioned that Julia does not impose any particular style of parallelism on the user but provides a set of blocks that allows implementing parallelism as programmer likes.

Julia is scientific oriented language and it influence on all aspects of language. There are built-in mathematical oriented data types in Julia, such as rational and complex numbers. Dynamic type conversion avoids a loss of accuracy, which is important in the scientific calculations. Also a large variety of built-in mathematical functions and a large number of libraries are present.

One of the main goals of the language was to achiev high performance close to the C/Fortran, and clarity interpreted languages. Besides compiling source files into an executable program julia has the ability to execute language constructs interactively (like interpreted languages), as well as an interface to support iPython.

Another key feature of the language is simple API for functions and subroutines of C/Fortran/Python. The standard library has lots of built-in wrappers for popular libraries (eg, BLAS and LAPACK). Also, there is a possibility to connect external libraries C / Fortran / Python.

Julia supports multiple programming paradigms: procedural, object-oriented, functional programming. Language is under active development and is replenished with libraries for various fields of mathematics, physics and engineering sciences.

1. The Julia Language — http://julialang.org/
2. The Julia Language·GitHub — https://github.com/JuliaLang