CLAVIRE: e-Science infrastructure for data-driven computing

Konstantin V. Knyazkov, Sergey V. Kovalchuk*, Timofey N. Tchurov, Sergey V. Maryin, Alexander V. Boukhanovsky

University ITMO, St. Petersburg, Russian Federation

A R T I C L E   I N F O

Article history:
Received 5 April 2012
Received in revised form 19 August 2012
Accepted 22 August 2012
Available online 29 August 2012

Keywords:
Distributed computing
Workflow
Domain-specific language
Data-driven approach
e-Science
Composite application

A B S T R A C T

The paper introduces CLAVIRE (Cloud Applications VIRtual Environment) platform. Architecture of the platform is shown with a focus on the abstraction which enables the integration of the distributed computational resources, data sources and the software. Coupled domain-specific languages EasyFlow and EasyPackage for unified workflow design are represented. Five classes of users’ interfaces are proposed as a basis for human–computer interaction support in CLAVIRE. Interactive workflow model is implemented as a prospective approach for data-driven composite applications.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Nowadays the scientific experiment often requires huge amount of computation during simulation and data processing. Performance of contemporary supercomputers is increasing rapidly. It allows to solve the computation-intensive scientific problems by processing large arrays of data stored in archives or produced by sensor networks. Thus today we can speak about a new paradigm for scientific research often called e-Science [1]. This paradigm introduces many issues (that have) to be solved by collaboration of ICT-specialists and domain scientists. As the paradigm is tightly related to processing the large arrays of data which are observed within the nature or produced by simulation software, it is required to develop new tools within data-driven approach (DDA) for arrangement of available resources for solving e-Science tasks [2]. Development of computational infrastructure for DDA-computing requires to integrate heterogeneous computing systems, ubiquitous sensors, imaging devices, and other data gathering devices, and to develop methodologies and theoretical frameworks for their integration in dynamic simulation systems [3]. Investigation of the abstractions which allows to integrate the distributed resources is an issue of especial importance [4,5] for the development e-Science infrastructure.

Contemporary computational tasks are characterized by the structural complexity: since they include many subtasks, they require different resources (software, hardware, data storages, decision making procedures, etc.) to be composed within one solution. Today one of the most popular solutions for joining distributed resources is the workflow (WF) approach [6] which permits to organize interaction between different resources presented as the services within computational environment. This approach was successfully applied in the number of e-Science infrastructures by means of WF-management systems (WMS) (e.g. [7,8]) and allows to organize interaction between different resources presented as the services within computational environment. Nevertheless in the frame of DDA having great diversity of resources of all categories (hardware as well as software or data resources) the problem of interoperability of the resources still remains.

This paper presents CLAVIRE (Cloud Applications VIRtual Environment) platform as the e-Science infrastructure platform for DDA-computing. The platform supports the high-level abstract description of computational processes in terms of composite applications, using a set of domain specific software and distributed data sources available within the service-oriented distributed computational environment. Composite applications of CLAVIRE are described using an abstract software calling (without definition of particular resources) which can be mapped on available hardware resources. It allows to run the software on different computation platforms (including environments like Grids or cloud infrastructures) using automatic scheduling procedure for resource selection.

* Corresponding author.
E-mail address: sergey.v.kovalchuk@gmail.com (S. V. Kovalchuk).

1877-7503/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2012.08.006
Knowledge-based tools decision support for the users during WF composing and execution is provided by means of the set of intelligent human interactions interfaces.

2. Backgrounds and architecture of CLAVIRE

CLAVIRE is built on basis of iPSE (Intelligent Problem Solving Environment) concept [9] which extends PSE concept [10] with the knowledge-based simulation support. The iPSE concept allows to hide the technical details of the used infrastructure: it is possible to interact with the system using the domain-specific languages which are automatically translated into the composite applications performed using the services available within e-Science infrastructure. This approach integrates different resources using domain-specific description of theirs usage. It enables to implement integration of heterogeneous computational and data resources and unified access to them. Moreover this approach allows to describe the composite applications in the form of abstract WF which are then translated into the calling of available services (concrete WF).

Fig. 1 represents the high-level architecture of CLAVIRE. Key features of the platform are as follows:

- Domain-specific language (EasyPackage) is used to describe available software packages and data sources, their calling procedure, parameter passing and used data formats. These formal descriptions are provided by CLAVIRE/PackageBase service.
- Abstract workflow description using domain specific language (EasyFlow) is used to define a composition application to be run. The translation of abstract WF description to the executable form is provided by CLAVIRE/EasyFlow core service.
- Interpretation of the domain-specific languages produces composite application, which calls computational services within cloud environment provided by Cloud Environment Management subsystem of CLAVIRE.
- Dynamic scheduling of the composite application allows overcoming performance issues using stochastic and deterministic prediction of computation time. This function is delivered by Sheduling and Execution subsystem of CLAVIRE on the base of performance models which are stored in the CLAVIRE/PackageBase service.
- Different computational resources are integrated within the single environment with unified access interface (single computers, Grids, virtual machines in IaaS clouds, see Computational Resources subsystem of CLAVIRE).
- High-level graphical user interfaces are presented for human–computer interaction within problem domain. These interfaces are provided by User Interfaces subsystem of CLAVIRE, see Section 4 for details.
- Knowledge processing allows to present the intelligent user support during whole process of simulation and data processing. The knowledge base of CLAVIRE (Knowledge storing subsystem) includes the expert knowledge of three main domains: (a) problem domain, which permits to make the domain-specific definition of WF; (b) ICT domain, which defines the knowledge on software and hardware usage, performance issues, data processing, etc.; (c) general simulation knowledge, which supports the composite application building process. All the knowledge is presented as the set of ontology structures.
- Integration with different data sources and visualization facilities within the DDA is provided by Scientific Visualization service (may be interpreted as local software).
- Further sections describe these features with more details.

3. Unifying of the resources and services

General issue of WF design for DDA-computing is based on the abstractions which allow to unify the descriptions of computational software and data sources in the frame of composite application. CLAVIRE provides the abstract WF design on the base of domain-specific languages (DSL) EasyFlow and EasyPackage.

EasyPackage – DSL for the description of software packages. This language allows the unification of software management process taking into account the diversity of technologies, hardware architecture, and folders structure being used. Structures of EasyPackage allows to describe the following information: input and output parameters of the software package, data formats being used for parameters passing, running commands, pre- and post-processing of the data, etc. As this information defines the whole process of calling software, usually performed by the user manually, it is possible to make this process automatic. Thus the CLAVIRE allows hiding particular software calling from the user. The platform converts the unified data into structures (files, command line arguments, environment variables) which permit to call the particular software. EasyPackage was developed on the basis of Ruby language, thus it supports all the structures of this language. Simplified language grammar is presented on Fig. 2.

EasyFlow – DSL for the composite application description in a form of abstract workflow. The language allows to define the composite application as a directed acyclic graph (DAG), containing the software calls (steps in terms of EasyFlow) with unified domain-specific parameters which were described earlier using EasyPackage language. Each step can define parameters explicitly or can receive them as the results of other steps. This defines data dependencies of the step. Apart from the data dependencies it is possible to define explicitly the control dependencies which identify the order of the steps. Thus data- and control-dependencies defines the edges of the DAG. Fig. 3 shows the simplified grammar of EasyFlow language.

EasyFlow and EasyPackage allow CLAVIRE to describe the composite applications as a WF containing calls of unified software services. Then the WF can be interpreted (mapped on the set of available resources) and automatically executed without any additional requirements from the user within cloud computing environment, see Fig. 1.

For DDA-computing which requires the real-time data gathering CLAVIRE provides the extended model of WF execution – Interactive WF (IWF), see Fig. 4. This model considers the subset of WF steps as the continuous runs with the interaction with other (also continuous or serial) steps allowed using channels – special type of data-dependency (communication dependencies). Also IWF model allows to integrate different sources for data being fetched within continuous running of WF, e.g., this can be useful for building any early warning system, which should react on data arrival [11]. In case of IWF execution process can be terminated manually or by special event raised.

4. Intelligent human interfaces

As the most of the WMS’s users are domain specialists (sometimes without strong ICT experience), the domain-specific human–computer interaction process with the high level of user support becomes one of the most important issues. Five classes of the users’ interfaces in CLAVIRE are introduced to support the human–computer interaction on different levels of abstraction (see Fig. 1).

(1) Console or program interface. Provide the lowest level of access to CLAVIRE using CLAVIRE/JobPusher service. It allows to run
Resource access

Cloud environment management

Computational resources

- Clusters
- Grids
- IaaS clouds
- Data resources
- Data source #1
- Data source #N

Workflow interpretation

WF-based interface

CLAVIRE/Ginger

Intelligent instructor

CLAVIRE/iKnowledgeTree

Domain objects

VirSimO

Knowledge editor

Scientific visualization

User interfaces

Console interface

CLAVIRE/JobPusher

Problem-oriented interface

POI

WF-based interface

CLAVIRE/EasyFlow

Domain-specific IT Simulation

Domain #1

Domain #NK

Cloud environment

management

Clusters

Grids

IaaS clouds

Software resources

Software #1

...

Data resources

Data source #1

...

Data source #N

Scheduling and executing

Software description

CLAVIRE/PackageBase

Ontological structure

- Simulation
- IT
- Domain-specific
- Domain #1
- ...
- Domain #N

Software resources

- Software #1
- ...

Data resources

- Data source #1
- ...

Data source #N

Workflow interpretation

CLAVIRE/EasyFlow

Intelligent instructor

CLAVIRE/iKnowledgeTree

Domain objects

VirSimO

Knowledge editor

Scientific visualization

User interfaces

Console interface

CLAVIRE/JobPusher

Problem-oriented interface

POI

WF-based interface

CLAVIRE/Ginger

Intelligent instructor

CLAVIRE/iKnowledgeTree

Domain objects

VirSimO

Knowledge editor

Scientific visualization

Fig. 1. Architecture of the CLAVIRE platform.

```plaintext
PackageDesc ::= {LibraryImport} (RubyClassDef) GeneralDesc {Section}
LibraryImport ::= use LibraryName
GeneralDesc ::= {DescFieldDesc} String
DescFieldDesc ::= name|display|vendor|url|license|description
Section ::= : SectionName "(" (ParameterDef ")")
SectionName ::= inputs|outputs|execution
ParameterDef ::= [public] ParameterType "(" {Field} ")"
ParameterType ::= param|file|file_group|models
Field ::= DataField|DataType|Procedure|DataProc
DataType ::= (name|display|description|depends|default|depends|package|filename|path|filters|coeffs) RubyExpr
Procedure ::= (required|validator|enabled|evaluator|expected|estimator)
Lambda ::= ["\{\" RubyBlockWithParameters \"]"
DataProc ::= (assembler|extractor) RubyClassDef Expression
```

Fig. 2. Simplified grammar of EasyPackage.

```plaintext
Application ::= {InputData} {WAttributes} {((Comment)}{BlockDesc})
Comment ::= "/\*" String
InputData ::= require IdsList ";"
WAttributes ::= "(" flow ")" ; Id "=" Expression ")"
BlockDesc ::= (Attribute) (step~step) Id runs CompositeId [after IdsList] ParametersList [PostProcessing ] ";" 
CompositeId ::= { Id [Point] }
IdsList ::= { Id ["," ] }
Attribute ::= "(" Id "=" Expression ")"
ParametersList ::= "; ")"
Parameter ::= Id "=" Expression
Port ::= Id "=>$" ObjectField
PostProcessing ::= post code ruby RubyCode code end
Expression ::= Literal | ObjectField
ObjectField ::= CompositeId [ ";\" Expression\"]
```

Fig. 3. Simplified grammar of EasyFlow.
the composite applications defined by using EasyFlow language with the direct posting of the workflow through this interface. Mostly this class of the user interfaces (UIs) is intended to be used by developers, who need to integrate CLAVIRE environment into their own solutions.

(2) Problem-oriented software interface. This class of UIs can be constructed using the formal software description which is provided by CLAVIRE/PackageBase service. The UI allows to define the parameters of calling the software, which then can be turned into WF, containing the software call through automatically generated web-interface. This class of UIs is intended to be used during solving the typical tasks which usually differs only by the set of parameters.

(3) Workflow-based interface. UIs of this class are developed to allow WF design and composition (from scratch or using templates). These UIs can use the textual or graphical notation of abstract WF (or both of them). Mainly the users of such interfaces are researchers who try to explore some natural phenomenon by simulation. Within CLAVIRE this class is implemented within CLAVIRE/Ginger service (see Fig. 5) which allows

---

**Fig. 4.** Interactive workflow execution.

**Fig. 5.** Workflow composition using CLAVIRE/Ginger UI: Left, input data and file manager; Center, EasyFlow code; Right, abstract WF (see Section 5).
to perform the workflow design using EasyFlow language, running, and fetching the results of running.

(4) **Intelligent instructors.** This class of interfaces is devoted to the comparison and estimation of solution (composite application) quality. With the use of expert knowledge it is possible to estimate the quality of solution by a set of metrics (e.g., performance, precision, reliability, etc.). As there can be a lot of possible solutions, this class of UIs should involve special graphical elements and structures for solution comparing, selecting, and estimation. In CLAVIRE platform this UI is implemented within CLAVIRE/KnowledgeTree software [12]. This software arranges all available kinds of WFs within a tree-based graphical structure which allow to explore and to estimate them according to set of metrics calculated by the analysis of presented data and selected tasks. Mainly such interfaces are developed for less experienced domain specialists (including students) who require support to build the composite application.

(5) **Domain objects.** This class of UIs is developed for the support composition of complex WFs mainly within multidisciplinary tasks. Domain objects are conceptual entities which aggregate models for simulation set of properties of corresponding natural objects. These models can be mapped (within particular scenario) into a set of software services calls, i.e., workflow can be automatically composed. As the domain object represents the domain-specific knowledge on the simulation process this approach allows (a) define structure of the system to be investigated by the selection and tuning appropriate domain objects; (b) run the simulation, automatically translate the structure into workflow; (c) present the results of simulation using the domain interpretable images which can be built using the defined structure. In CLAVIRE this interface was implemented within VirSimO service which operates the graphical editor of system's structure using the library of available domain object. After its construction the system's structure is translated into the workflow which can be edited using CLAVIRE/Ginger UI. Example of using this type of UI is described in details in Section 5.

5. Execution process with DDA support: simulation of marine structures

Within this section the more complex example is considered in the frame of DDA-approach. The problem is the numerical exploration of extreme dynamics of the coupled marine objects in a rough sea, using the real-time data from the correspondent measurements in model tank. In the frame of abstraction level which we use in CLAVIRE, four domain objects are to be simulated using several packages integrated to CLAVIRE/PackageBase:

(1) Rough sea. Spectral characteristics of waves are simulated using SWAN software [13], which is run with the input driving wind calculated by local software using the aloft wind forecast.

(2) Two ships: one (Ship#1) driven by own power, and another (Ship#2) driven by first one. The ships dynamics is simulated according to the waves produced by the first object using ShipXDs software [14].

(3) Rope, connecting two ships and transferring forces form one ship to another. The rope is also simulated by local software.

Each of these four objects can be presented by corresponding domain object. Fig. 6 shows these objects related as it can be done using domain objects concepts. Thus this system's structure can be constructed using UIs of fifth class (see Section 4).
This example includes the external data sources related to the model tank experiment, e.g., generated WF can be connected to the model tank as a data source. In this case a part of the data (spectral parameters, wave elevation, roll and pitch angles, and rope tension) can be received from the real time measurements. Possible scenario can be as follows: (1) wave characteristics are simulated by WF; (2) waves are generated in the tank using the simulated spectral parameters; (3) model of Ship#2 is driven using a rope by the trolley with the constant speed observing the forces produced by the rope tension; (4) behavior of the Ship#1 being simulated using the waves parameters and measured rope forces; (5) (optional feedback) results of Ship#1 behavior can produce impact on the forces taken into account on step 3. Fig. 7 shows the possible implementation of this extended scenario within interactive WF (see Variant 2 usage in comparison to Variant 1, which represents a complete simulated scene). Steps of the scenario are depicted with the numbers on Fig. 7.

The last step (feedback) is possible (a) only for interactive workflow; (b) in case a platform can process IWFs in real time. First requirement is fulfilled using CLAVIRE implementation of IWF. As for the second – experiments shows that CLAVIRE implementation has rather low latency level (about 80 ms), thus it can be used in such scenario.

In Fig. 8 the snapshots of scientific visualization of the above mentioned example are shown. All the structures for 3D scene were produced through CLAVIRE Scientific Visualization service and visualized by means of local software. The features of IWFs model allow not only high-quality real-time visualization, but also interactive control of the computations [14,15].

6. Conclusions

CLAVIRE introduces the high level toolbox for solving e-Science tasks within data-driven approach. Being based on iPSE concept it allows to interact with a user being supported by domain-specific knowledge. Main advantages of CLAVIRE are as follows:

- It allows to unify the access to the resources and to integrate them within the high-level domain-specific composite application using coupled DSLs for software description and WF definition.
- CLAVIRE platform provides the user with a set of UIs (implementing corresponding classes) which permits to access the simulation environment on different levels of abstraction.
- It has the ability to run the interactive workflows as well as more common batch workflows in a common way hiding whole complexity of cyber infrastructure from the domain user.
- Scheduling system of CLAVIRE optimize the execution time using parallel runs, automatic parameters tuning (using performance models) as well as the solution selection according to performance estimation.
- Within DDA the platform allows to integrate the different data sources into simulation workflow (including IWFs) which enables the building of a wide range of data-intensive composite applications using CLAVIRE.

Acknowledgments

This work was supported by the project granted from Decree 218 (computational infrastructure) and 220 (interactive WF model) of Government of the Russian Federation.

References