Dynamic Domain-Specific Language for BigData Tasks’ Description

Artem Zakharchuk¹, Sergey Kovalchuk², Denis Nasonov³

Abstract: The paper presents a dynamic Domain-Specific Language (DSL) which is developed to provide the capability of high-level BigData task descriptions within e-Science applications. The dynamic structure of the DSL supports language structure extension depending on a particular problem domain defining specific requirements, data processing, and aggregation and simulation procedures. The extension is implemented using the set of domain-specific libraries providing integration into the DSL interpretation system. The developed DSL is integrated with the cloud computing environment CLAVIRE and DSLs’ family used to describe composite application, software packages and resources within this environment. Together with knowledge-based technologies implemented on the basis of this environment (Virtual Simulation Objects - VSO, knowledge-based languages, etc.) the DSL is used to develop and execute workflows within a distributed cloud environment combining computational-intensive tasks with data-intensive MapReduce processing within common semantic space defined by domain knowledge.

Keywords: domain-specific language, big data, map-reduce, cloud computing, code-to-data

1 Introduction

Contemporary e-Science tasks often require data-intensive applications to be incorporated as a major part of the composite solution [He09]. Today a huge amount of various data (stored in the data warehouses, accessible as observation of forecasting services, real-time monitoring data, etc.) is available for scientific analysis, information retrieval and knowledge discovery. The growth of volume and variety of available data together with processing performance requirements causes the emergence of the BigData area [PZ14]. The BigData area collects a set of practices, methods, technologies which are aimed to manage better the large data arrays. For example, the MapReduce model [DN14] is considered today as a standard solution for distributed processing of large data arrays. Nevertheless BigData still has a lot of issues to be solved. One of the questions currently discovered within the area of BigData is high-level user-friendly task definition [As14]. A common way of high-level task description is Domain-Specific Languages (DSL) like Pig (https://pig.apache.org/) or Hive (https://hive.apache.org/). Moreover the DSL can serve as a core technology for solving data and operation fusion problem combining high-level task description with performance optimization (see e.g. [Ac12]). Nevertheless complex e-Science tasks often require high-level integration of diverse data

¹ ITMO University, Saint-Petersburg, Russia, zakharukart@gmail.com
² ITMO University, Saint-Petersburg, Russia, kovalchuk@mail.ifmo.ru
³ ITMO University, Saint-Petersburg, Russia, denis.nasonov@gmail.com
and software (often available within a distributed environment) which makes a high-level task description more complicated. Within the e-Science area the common approach to description of the complex domain-specific tasks is workflow (WF) [Gi07]. Still the WF concepts usually exploit the data-to-code approach (data is transferred to the executable services) when the BigData solutions are usually based on the code-to-data concept (executable modules are transferred to the data sites) [Ma10]. The differences in the approach make it difficult to integrate seamlessly the BigData solutions into the WF management systems (WFMS): the existing solutions usually have a strong conceptual border between these two conceptual approaches, e.g. by considering the BigData subsystem as a special kind of service integrated into WF [BBB13]. On the other hand, nowadays e-Science solutions tend to deviate from pure WF concept (as mostly a programming concept) and shift towards the approaches more friendly to end-users (mostly domain scientists) who often have a lack of programming background [Mc09]. Moreover today’s trends require the e-Science solution to be enhanced from procedural WF description towards complex system-level investigation [FK06].

Within the presented work we try to develop a solution which enables the end-user to define high-level tasks which combine seamlessly both data-to-code and code-to-data approaches by the use of knowledge-based support for simulation-based scientific exploration [SKB13]. As a core part of the solution we have developed a dynamic domain-specific language (DSL) which can adapt to the particular problem domain and might be integrated into the cloud computing environment to support the BigData tasks’ definition. The paper presents the main features of the developed language and its implementation for composite application development.

2 Solution Development

2.1 Code and Data Relationship

The code-to-data and data-to-code approaches consider two artifacts usually involved into application development. Within our approach each of these artifacts can be either provided by the user (considered as portable) or provided by a reference to remote site (abstract definition possible). This permits us to identify four situation and popular solutions related to each (see Table 1).

The presented features and systems are mentioned to highlight diversity of the approaches which are used within different e-Science applications. Within the presented work we consider a task as a high-level description of the problem which contains or can be interpreted into provided data (PD) and pieces of code (PC). To solve the task, the distributed environment, which contains computational services (LC) and data stored within distributed storage or available on demand using remote services, were used.
Provided code (PC) | Linked code (LC)
---|---

Tab. 1: Code and data processing

The distributed data, available for processing can be considered as informational resource provided to the user on demand (where the request is formalized within the provided code). This point of view considers the processing of BigData within the scope of cloud computing [Fo08] approach. Mapping the existing cloud computing Maturity Models (CCMM) (see e.g. [Gtsi]) which describes the level of cloud computing concept’s implementation onto the data processing task can define several stages of data provision: **Consolidation.** Having a lot of data sources, a large volume of data storage could be formed and updated using external services, public data warehouses, simulation archives, etc; **Abstraction.** Unified access to the data with different formats can be provided by solution which supports abstract data types by automatic mapping of data requests onto the available data stored or provided as a service; **Automation.** Use of the MapReduce model requires from the user to develop an executable module. As the typical user of e-Science solution usually has a lack of technological background, the automation level should provide high-level expressive tools for task definition; **Utility.** This level should support the user with extended analytics toolbox statistical aggregation, data mining, etc; **Cloud.** Finally the data become the subject to operate in a market way where the cloud system automatically processes the request from the user providing the results of the task solving and hiding all the low-level procedures.

The CCMM levels can define the key waypoints during the development of solution architecture which can serve as a high-level e-Science tool to efficiently support the user during the scientific investigation.

### 2.2 BigData within e-Science Tasks

Considering the typical e-Science tasks, BigData requirements appeared within such tasks, and CCMM level mentioned earlier, the following features can be defined for the developed solution:
- **High-level semantic description** of available data to integrate various data sources and to provide the user with capability to work with abstract data types defining them with domain-specific semantics.

- The semantic description should link the data with the *structure of system* being analyzed and thus support the system-level science investigation.

- Generally the e-Science tasks are solved within Simulation-Driven Approach (SDA). The data processing should be *integrated with the simulation procedures* a) by being incorporated into the hybrid high-level composite application (inferred from the high-level task definition); b) by performing simulation procedures in code-to-data way within the data storage.

- **Domain-specific languages** (DSL) [DKV00] provide a powerful toolbox for high-level interaction with the user. Still usually the domain-specific language is developed for the specific problem.

- Considering the e-Science area a huge variety of domains and tasks can be identified. Thus the developed DSL should be *dynamically adoptable* to the particular problem domain and task.

### 2.3 Solution’s Architecture

The proposed solution is based on the idea of knowledge-based expressive toolbox [Ko13] where the hierarchy of languages (with textual or graphical notation) is defined by the set of knowledge from different problem domains. The basic hierarchy of expressive tools includes five levels: L1 – services description; L2 – services composition (WF); L3 – simulation objects description; L4 – objects composition (description of investigated system; L5 – high-level task definition.

To extend this architecture the parallel tier (see fig. 1) is presented which describes the code-to-data part of composite application built during high-level task interpretation. To implement this tier the dynamic DSL is developed which serves to describe the high-level BigData tasks. It incorporates the imperative procedures, defined by the task being solved and applies them to the structure of investigated system. As the structure is linked with the available data, the interpretation of the language can automatically generate the full executable MapReduce code which is ready to run on the data storage system. The code-to-data part of the composite application is interconnected with the data-to-code part in several ways: a) the system’s structure is shared between these two parts and used both to generate general-purpose executable WF and to select the proper data during data analysis; b) dynamic dataset, which is updated during simulation and data analysis, is sharable which enables to join the parts and apply them separately; c) distributed data storage can be used to transfer small datasets to the computational services (in data-to-code way); d) finally the cloud management system can control the deploy of portable software to the data storage nodes to support short local runs for the simulation purposes (see the requirements).
The dynamic DSL being developed within the knowledge-based approach incorporates the following artifacts:

- **Domain-specific semantics.** A set of domain-specific objects which are used to describe the structure of investigated system is defined. This part of knowledge enables interconnection with WF and simulation procedures on the levels L4-5.

- **Data formats.** To support the integration and high-level definition of the task, the semantically marked data of format description and atomic analysis procedures should be described.

- **Data aggregation patterns.** To interpret imperative description of procedures within the high-level task with MapReduce code generation, the data aggregation and statistical analysis procedures should be available.

- **Cloud infrastructure knowledge** should be provided to support a) further processing of the results by the other parts of composite application; b) call the local software described in the same way as regular cloud computing services.

## 3 Implementation details

The developed solution is based on CLAVIRE [Kn12] cloud computing environment which actively used knowledge-based technologies for solving-Science tasks. It supports a set of DSLs for description of available software services (EasyPackage language based on Ruby) and develops high-level composite applications in form of abstract WF (EasyFlow declarative language). Additionally the Virtual Simulation Objects (VSO)
technology was developed to support high-level semantic description of investigated system which can be mapped on AWF structure and executed within the cloud environment [Ko12]. The platform uses distributed data storage (DStorage) which was extended to support MapReduce operations (described as Java code) on the stored datasets [KRS13]. Within the presented research we developed a dynamic BigData DSL which serves as an expressive tool for definition of high-level data analytics tasks by the use of the mentioned technologies. The goal of the language is to extend the implemented CLAVIRE-based hierarchy of languages to support the development of hybrid composite applications exploiting both data-to-code and code-to-data approaches. The DSL was developed on the basis of ANTLR which is cooperated with Java environment and provides powerful features for DSL development. The following subsection describes the developed solution with more details.

3.1 DSL Structure

Current version of the dynamic DSL is being developed for CLAVIRE platform, in particular for execution within distributed storage (DStorage). The basic architecture of the distributed data processing within CLAVIRE and DStorage is presented in the fig. 2.

The data processing solution implements fixed MapReduce scheme which allow us to process data files stored on the nodes of the distributed storage. The data processing module is implemented by Java programming language and uploaded to the storage with automatic replication onto the whole set of storage nodes. In order to run data processing task a request has to be processed by the core of data storage which accepts configuration and additional parameters (query strings, additional data etc.) and runs distributed processing of files with aggregation of the results. The distributed data processing solution is used as a basic technology which is extended by the means of DSL integration. Within the storage architecture the DSL script is processed as a parameter of distributed task while the DSL interpreter is implemented as a data processing module and uploaded to the storage in a general way.

In order to support semantic integration and high-level concepts’ semantics the VSO technology is used as one of the extensions of the CLAVIRE platform. The VSO concept and technology provide the high-level semantic abstractions which integrate domain-specific objective description of investigated system and simulation infrastructure. The integration with CLAVIRE enables the use of the abstract WFs in a form of extended EasyFlow (DSL for WF description) (grammar is shown in fig. 3) and high-level domain-specific parameters of available software packages described using EasyPackage (DSL for abstract software services description).

In MapReduce scheme a map procedure defines processing of files within the storage, while reduce procedure defines aggregation of the mapping results. As the data is distributed, the analysis functions may be applied to them only when certain conditions are fulfilled. These conditions should be described in the domain library, which is dynamically linked to DSL.
Dynamic DSL for BigData Tasks’ Description

Fig. 2: Distributed data processing within CLAVIRE platform and query interpretation

Exactly these libraries determine the dynamic of the DSL. The domain library should contain the following items:

- The atomic structure of data stored in files.
- Constructed data structures are generated during data analysis.
- Unified algorithms for reading data structures from the storage
Algorithms for processing and analysis of data structures (1, 2).

Also, during execution of data processing, the various software packages should be able to run. It is constructed by using EasyPackage description within local PackageBase and local ResourceBase of CLAVIRE platform.

Simplified grammar of developed dynamic DSL is shown in figure 4.

```plaintext
Application ::= (InputData) Select [Create] [Simulate] Out
InputData ::= Id ::= "value"
Select ::="select" Id "where (" (Condition) ")"
Create ::= "create" Id "from" Id "where (" (Condition) ")"
Simulate ::= "simulate" ("VSO") "Package" Id "from" Id ["(" IdList ")] ParameterList
Out ::= "out" outParam ("=" outParam)
Condition ::= Id sign Expression
IdList ::= (Id)
ParameterList ::= Id ::= "Expression"
OutParam ::= Id "=" Id
```

Fig. 4: Simplified grammar of developed DSL

### 3.2 Query Interpretation

Query written with the DSL and described with a specific problem analysis and processing data, is transmitted together with other parameters at startup of data processing task. This script should consist of keyword of the DSL and domain-specific keywords. Procedures Map and Reduce for distributional data processing is generated under the condition of this request. Processing the DSL query is shown in fig. 2.

Currently the DSL contains the following keywords: **Select.** Creates instances of the objects, which are described in the domain library and fill with data from storage using reading algorithms; **Where.** Analyzes the object using algorithms from the domain library; **Create.** Creates the object from other (objects); **Simulate.** Executes software packages or VSO by using Local PackageBase and Local ResourceBase; **Out.** Presents the results.
In general, the skeleton of query script is shown in fig. 5. Here beside the basic DSL keywords the following structures are presented for the purpose of dynamic extension of the language structure: a) references to semantically defined objects, parameters and models are provided using VSO technology or software packages for local calls defined using EasyPackage within the CLAVIRE environment; b) objects, procedures and conditions provided within domain libraries; c) data file semantic identification which identify the datasets within DStorage.

4 Hybrid Workflow Case Study

Experimental studies in cyclone’s analysis have significant importance for protection of Saint-Petersburg area from storm surges [AK07]. One of the vivid examples of such importance is the hybrid composite application that is described in the Listing 1.

**Listing 1.** Hybrid composite application for cyclones’ analysis

```plaintext
require BSHFile, projects, inControlPoints, points, searchedhirlam;
script CycloneSearch runs `BDDSL`

```dir = "/data/path", tag = "ERA"
``` :

```select Cyclone where (trajectory_path = "((65.67, 8.19), (65.97, 23.45), (60.97, 31.45))", accuracy = "(5, 5)"
``` out Cyclone.Parameters() as outs
```;
```

step CycloneStatisticalAnalysis runs `scilab`

```input_folder = CycloneSearch.Result.outs["data/out"],
script_name = "SCI_stat.sce"
```

script FloodSimulation runs `BDDSL`

```dir = "/data/path", tag = "HIRLAM",
config = CycloneStatisticalAnalysis.Result.outs["output.dat"],
points = points, projects = projects, BSHFile = BSHFile, searchedpattern = searchedhirlam, inControlPoints = inControlPoints
``` :

```select Forecast where (pressure_field = searchedpattern)
``` simulate package cyclonegenerator with parameters

```inConfigFile = configFile
forecastField = Forecast.field
```

```simulate VSO FloodSim from cyclonegenerator.Result [points, projects, BSHFile, inControlPoints] to FloodSimulation
```

out FloodSim.Result as out
```;
```

step FloodStatisticalAnalysis runs `scilab`

```input_folder = FloodSimulation.Result.outs["/out/"],
script_name = "FLOOD_stat.sce"
```

```15```
```15```
During the first step all cyclones that match to the provided criteria on trajectory path and its accuracy are detected. In the composite application this task is executed in the step CycloneSearch, that, firstly, extracts all the cyclones that are stored in ERA (http://apps.ecmwf.int/datasets/data/era40-daily/) files (operation "select"). Then a filtration (operation "where") of the found cyclones occurs, using parameters of trajectory path and accuracy that are extracted with the function Parameters(). All these actions are performed by using domain library Cyclone on the distributed storage nodes.

Statistically estimated parameters for the most probable cyclones’ emersion are calculated at the second step. For this purpose the step CycloneStatisticalAnalysis evaluates scilab package (software for numerical computation) in the general way (data-to-code).

During the third step all forecasts that are similar to the one provided by the user are detected and the cyclone with the estimated parameter is simulated. The step FloodSimulation is also executed in the distributed storage. Firstly, the forecasts, which comply with the conditions on the provided pressure (file "serchedhirlam"), are selected from HIRLAM (http://www.hirlam.org/) data. This part of the step uses the domain library Forecast. Secondly, package cyclonegenerator is executed locally on the storage agent, with the parameters from "configFile" file that was obtained in the previous step CycloneStatisticalAnalysis. The result of this package is transmitted to the VSO SeaForecast with other input data. Due to VSO two packages SWAN (wave model, http://swanmodel.sourceforge.net/) and BSM (Baltic Sea Model, generate water level forecast) are running subsequently on the storage agent. Finally water level forecasts are returned to the step FloodStatisticalAnalysis for analysis.

![Fig. 6: Forecasted water level in Saint-Petersburg for generated cyclones](image)

On the last step of the analysis of all water level forecasts is performed in FloodSimulation. This package is executed in the general way (data-to-code).

The results of completed composite application are presented on figure 6 (the plot was generated automatically by the last step of the application). In spite of the fact that on all found forecasts the cyclone with the same parameters was generated, the form and the height of the expected flood pick is different. It can be explained by two main reasons: pressure field influence by itself (part without cyclone) and simplicity of cyclone
Dynamic DSL for BigData Tasks’ Description

generation algorithm (it doesn’t take into account).

5 Conclusion

The dynamic DSL being developed provides a flexible expressive tool for high-level definition of data analysis tasks. The knowledge-based technologies used to interpret the DSL structure enables the interpretation of the terms and structures defined by the semantics of particular problem domain (or multiple domains in case of multidisciplinary tasks). The DSL is integrated with cloud computing environment CLAVIRE and enables the development of hybrid solutions by integrating the DSL with languages for WF definition. The complex system for high-level definition of the tasks allowing the automatic building of hybrid composite applications is now being developed on the basis of the presented DSL technology integrated with the existing expressive technologies of CLAVIRE platform.

Acknowledgements. This paper is financially supported by Ministry of Education and Science of the Russian Federation, agreement #14.578.21.0077 (24.11.2014). Data management facilities were developed within the project “Big data management for computationally intensive applications” (project #14613).

References


[DN14] DOULKERIDIS, CHRISTOS ; NORVÅG, KJETIL: A survey of large-scale analytical query processing in MapReduce. In: VLDB Journal Bd. 23 (2014), Nr. 3

A. Zakharchuk et al.


[HTT09] HEY, TONY; TANSELY, STEWART; TOLLE, KRISTIN: The Fourth Paradigm: Data-Intensive Scientific Discovery Bd. 99 (2009), Nr. 8 — ISBN 9780982544204


1218