A Task Management in the Intranet Grid

Petr Lukasik and Martin Sysel

Department of Computer and Communication Systems
Faculty of Applied Informatics
Tomas Bata University in Zlin
nam. T.G. Masaryka 5555, 760 01 Zlin
CZECH REPUBLIC
plukasik@tajmac-zps.cz, sysel@fai.utb.cz
http://www.fai.utb.cz

Abstract. The main purpose of this work is to explain the management and distribution of tasks in a grid middleware. The aim is to propose an environment that enables designing the batch jobs that use the standard software and system resources for communication and data exchange. The article explains JSDL specification for defining and management of one batch jobs. The motivation is to design a tool for an easy job definition. The result is a tool that does not require special programming objects for solving a specific task. This work deals with the mechanisms for monitoring and gathering information about the result of each processing task. The JSDL definitions provide the mechanism for the solution of these problems. Grid Scheduler more easily recognizes the status of a specific task.

Keywords: Grid, JSDL, POSIX, fault tolerance, return code.

1 Introduction

Planning and distribution of tasks are essential elements of a grid services. A tool that allows easy definition of the role and its distribution in the environment is a prerequisite for high-quality and user-acceptable Grid Services. The user should have a freedom as well as resources to easily tracking of their own processing. An important feature is that the Grid service has the least restrictive conditions for a successful job execution. (Type or version of software, operating system and hardware features). The user of the grid should to have a certain freedom. Not to be tied up of restrictive rules, except the rules relating to information security and data processing. The POSIX interface provides extensive options in the use of standard programs for communication and distribution of batch job. It also provides excellent portability of applications in various types of operating systems. This feature is convenient for defining the tasks in a grid.

2 The Job Distribution in a Grid Environment

JSDL (Job Submission Definition Language) is an XML-based computational specification for the management and distribution of batch jobs in a Grid

R. Silhavy et al. (eds.), *Software Engineering in Intelligent Systems*,
Advances in Intelligent Systems and Computing 349, DOI: 10.1007/978-3-319-18473-9_8

environment, developed by OGF JSDL-WG [1][2][3]. A current version 1.0 (released November 7, 2005) has also the definition of the POSIX support. This allows implementing the requirements described above. Important is Open Grid Forum support. JSDL should be considered as a standard [6].

The life cycle of the task instance represents the current state of the program which is currently executed (or pending to be executed). Each instance of the job it has a direct impact on the overall result. Status of the tasks that provides the Grid client is very important for the scheduler. The scheduler is responsible for the successful solution of the assigned task. Whereas it applies that in a number of fault conditions, this information is not delivered to the planners [5].

The same mechanism of the lifecycle also applies to the run of the whole batch job. Failure of a single part means a bad result of the entire job. The grid scheduler has to keep track of each state of all processed instances of the job. Scheduler based only on all available information, can successfully manage the processing tasks and to resolve error conditions. The states that can occur in the life cycle flow of the job are described in the fig.[1]. The task progress can take three states. These states can be categorized into two groups. The first group is the state when the task will return result of the processing (correct or incorrect).In the second group, cannot send a return value. Example is violent interruption of the task or power failure in the some parts of the infrastructure.

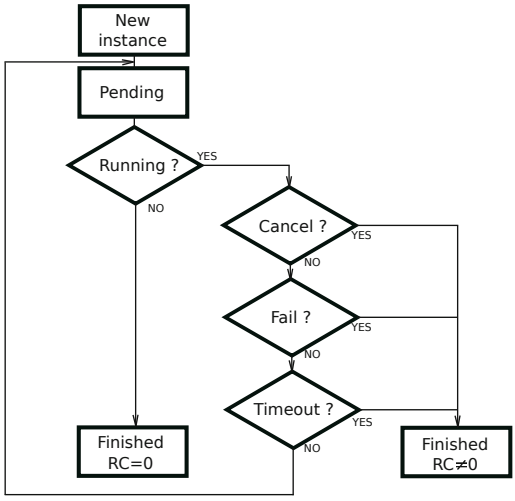

Fig. 1. Task and return code management

Grid scheduler has to solve the state, if the work cannot send a return value. The principle is as follows. Scheduler receives information about the interruption of the communication channel. If this information is not available, the planner has to monitor timeout, which is set as a parameter to the task, which is currently running. After a timeout, the task instance is declared as lost. This means

that an instance of task must be rescheduled again and sent back to the processing. To specify a timeout value is relatively difficult. Usually effective processing length, from which the value can be inferred is not known. Correct timeout can be specified based on some experience with specific job. There is, of course, possibility that the problem causes the task, currently running that do not have correctly handled error conditions (memory leak, divide zero). This task can crash the entire grid infrastructure. Therefore, the grid scheduler must solve this situation as well, according to predetermined rules. Job Submission Definition Language includes support that allows you to define ranges of computing resources (length of treatment, number of running threads, disk space). The task does not run if any of the parameters is exceeded. You can also define rules for communication with Grid scheduler while processing the return codes and rules in case of failure of one of the running instances.

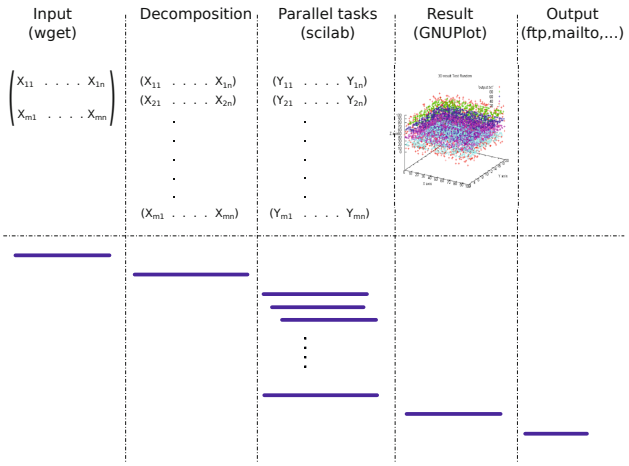

Fig. 2. Lifecycle of the job and task

3 The Standard Programs for Communication and Task Distribution

POSIX interface and its features are used for the distribution of tasks, data transfer and communication between the various components in the grid. This interface provides portability of program objects and commands on the system console. This allows the use of existing software. Grid user therefore does not need to write a special programs or objects to solve their task and enables the use of standard software (Python interpret, Mathworks, Scilab). The advantage is a choice of suitable existing components and using the standard and well-known user environment. See fig.[2].

This solution is a perspective in the local environment, such as on the university or corporate grid, which guarantee a certain standardization of software and system tools. The example below defines this job. Service wget retrieves data from various data repositories (http). The GnuPlot program generates a graph. The result is sent to the user via email. The return value is sent to the grid scheduler using the Web services or the Intranet Grid services fig.[3]. This particular example shows that the task does not require special software, but will allow the user to utilize the standard software. The JSDL has a definition for single-program-multiple-data (SPMD) parallel techniques. This extension supports various MPI environments. (MPI, GridMPI, Torch / MPI, MPICH). Parameter Sweep definition to the JSDL is also available for a parallel tasks. This extension defines rule to explicitly submitting various number of individual tasks of the same base job template. One definition allows generating a large number of parallel executable tasks [4].

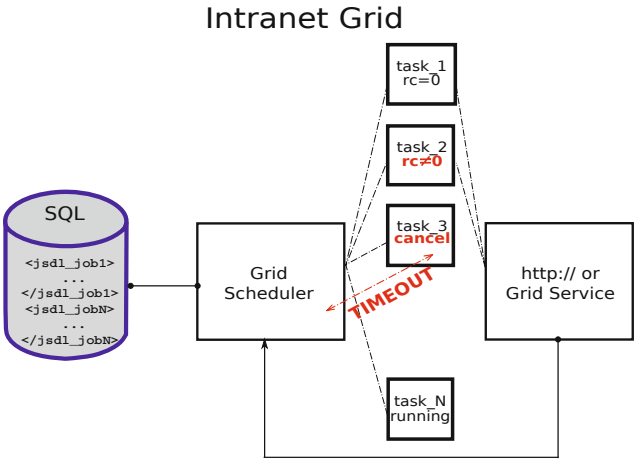

Fig. 3. Management of return values

4 Principles for Resolving of the Error States

The weak part in communication of the grid scheduler and grid agents is a violent termination of the currentl running job. Fatal error of the application causes a loss of information about the job. Grid scheduler does not get back information about this error. Fault Tolerance subsystem and its properties has a major impact on the reliability of the entire grid services. The challenge is to increase tolerance to the system failures, and recognize various error scenarios. Based on the information available, choose the correct recovery procedure. JSDL specification enables the user to define some limit values. This increases the intelligence of the system and also allows to better evaluating error conditions.

The basic limit values of the batch jobs is memory size, file size, number of threads and maximum run time. Exceeding some of the limit values indicates incorrect job status. Type of the limit value that was exceeded, allows solve to better the error status of the job. For example, invoke the rollback or send a job to the processing again.

5 Tools for the JSDL Support

G-Eclipse Plugin extends the Eclipse Project `eclipse.org` and provides support for Grid services. This environment provides services independent of a specific grid middleware. Plugin architecture enables developers to extend the g-Eclipse on a new feature. Example is integration of another middleware. Support for the gLite and GRIA middleware is also completed [6].

5.1 g-Eclipse - Support for Users

- *Grid user:* Has no detailed knowledge of Grid technologies. Allows you to run and monitor the progress of a job.
- *Grid operator:* Has a detailed knowledge of the Grid infrastructure. Operator has support for the management of local resources, and also supports external resources so called virtual organizations, where operators are the members.
- *Development engineer:* Expert on programming in the Grid applications. Developer has tools to develop, debug and deploy the application. Graphical editor for generating JSDL file is also available. Editor supports POSIX and Parameter Sweep extension.

5.2 A Program for the JSDL Parsing

The parser was designed for the Intranet Grid Middleware[7]. We use the Apache XMLBeans library and DOM model. The DOM advantage is parsing of XML document in the memory - no I/O operation is required. The aim was to determine the performance of the parser.

It has been shown the low requirements of the JSDL tools to system resources. Two types of tasks were measured. The simple Sweep-loop in the first task, (see the fig.[4] *sweep(1)* label) and the triple Sweep-loop in the second task, (fig.[4] *sweep(3)* label). There were generated the scripts in the number from one to 10000. Five measurements were performed for each job, and was carried out on two systems with different performance. It was verified by the linear increase of the time and a low system load, inspite of the XML based JSDL. In the fig.[4] is only average value presented. Detail of the measurement is in the 1 presented.

Table 1. Results for the sweep loop (1) and sweep loop (3)

sweep(1) M420

number of the jobs	time[s]
1	0.555 0.555 0.554 0.557 0.558
2	0.559 0.558 0.564 0.559 0.560
5	0.537 0.533 0.536 0.536 0.537
27	0.606 0.606 0.608 0.604 0.606
256	0.842 0.830 0.810 0.828 0.819
2000	2.067 2.070 2.142 2.056 2.030
13824	9.218 9.088 9.209 9.233 9.177

sweep(3) M420

number of the jobs	time[s]
1	0.553 0.559 0.550 0.548 0.548
10	0.559 0.557 0.600 0.556 0.566
100	0.672 0.661 0.679 0.676 0.706
1000	1.445 1.425 1.404 1.454 1.489
10000	7.412 7.524 7.438 7.501 7.625

sweep(1) T1700

number of the jobs	time[s]
1	0.156 0.155 0.158 0.157 0.156
2	0.160 0.160 0.175 0.158 0.156
5	0.149 0.152 0.150 0.151 0.151
27	0.168 0.170 0.170 0.170 0.170
256	0.233 0.230 0.228 0.228 0.226
2000	0.595 0.552 0.571 0.539 0.482
13824	2.613 2.355 2.591 2.546 2.482

sweep(3) T1700

number of the jobs	time[s]
1	0.155 0.152 0.155 0.156 0.151
10	0.158 0.158 0.158 0.157 0.157
100	0.197 0.193 0.193 0.191 0.190
1000	0.457 0.497 0.483 0.506 0.455
10000	2.534 2.607 2.504 2.590 2.493

5.3 The XML Schema of the Sweep(1) Loop

```
<!-- SWEEP(1) - job_NNNNNN.sh -->
<!-- LEVEL1.1 -->
<sweep:Sweep>
  <sweep:Assignment>
  .
  </sweep:Assignment>
</sweep:Sweep>
```

5.4 The XML Schema of the Sweep(3) Loop

```
<!-- SWEEP(3) - job_XXYYXX.sh -->
<!-- LEVEL1.1 -->
<sweep:Sweep>
   <sweep:Assignment>
   .
   </sweep:Assignment>
   <!-- LEVEL2.1 -->
   <sweep:Sweep>
      <sweep:Assignment>
      .
      </sweep:Assignment>
      <!-- LEVEL3.1 -->
      <sweep:Sweep>
         <sweep:Assignment>
         .
         </sweep:Assignment>
      </sweep:Sweep>
   </sweep:Sweep>
</sweep:Sweep>
```

6 Result

The text describes the use of JSDL generator as a batch jobs. The primary objective was to take maximum advantage of the standard and existing software and system services. This allows the user, to be more creative in the design of task. At the first glance, it may also appear that JSDL brings some complications

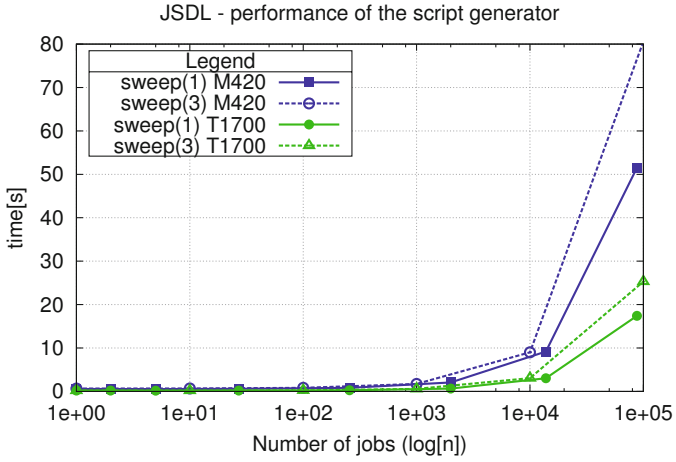

Fig. 4. Sweep parm and system load

Table 2. A list of workstations where measurements were made

A	**List the hardware where it has been JSDL parser tested**
1	**DELL T1700 64 bits**
Memory	16GB RAM
CPU	Intel(R) Xeon(R) CPU E31225 @ 3.10GHz 64 bits
OS	Ubuntu 14.4 LTS
Java	OpenJDK 1.7
2	**Fujitsu Siemens Celsius M420 32 bits**
Memory	1.96 GB RAM
CPU	Intel(R) CPU E 6750 @ 2.66GHz 32 bits
OS	Ubuntu 12.4. LTS
Java	OpenJDK 1.6

(XML parser, job generator). The advantage is obvious when it is necessary to generate and manage thousands of concurrent instances of a single task.

Another direction for development of intranet grid middleware will be design and use of JSDL for applications in the Apache Hadoop cloud. Hadoop is the executive and promising platform. Problem of forcibly terminated task is solved by using redundant processes.

7 Conclusion

This work is focused on the job definition in Grid computing environment. The Grid usually have a large amount of heterogeneous sources that have many different configuration and many different internal rules. JSDL has a set of instructions that unify these rules. JSDL has the tools, for example G-Eclipse Plugin that facilitate definition of the tasks. Users do not to have detailed JSDL knowledge. Measurements that have been made not demonstrated great demands on the computing performance of the parser. The advantage JSDL is easy definition of parallel tasks. Parameter sweep easily defines the number of jobs and their identification.

Comparing JSDL and similar technologies, which are used to define the job in grid computing, for example JDL (Job Definition Language), we came to this conclusion. JSDL is suitable for the commercial applications that require easy design and definition of the task.

JDL is a Python-based language for describing jobs.[9] It requires an experienced user. Provides more possibilities for the definition of the job. Deployment JDL is preferable in the research centres or universities.

References

1. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: GFD-R.136 Job Submission Description Language (JSDL) Specification (July 28, 2008), http://forge.gridforum.org/projects/jsdl-wg, Copyright (C) Open Grid Forum (2003-2005, 2007-2008). All Rights Reserved
2. Humphrey, M., Smith, C., Theimer, M., Wasson, G.: JSDL HPC Profile Application Extension, Version 1.0 (July 14, 2006) (Updated: October 2, 2006) Copyright Open Grid Forum (2006-2007). All Rights Reserved
3. Drescher, M., Anjomshoaa, A., Williams, G., Meredith, D.: JSDL Parameter Sweep Job Extension Copyright © Open Grid Forum 2006–2009. All Rights Reserved (May 12, 2009)
4. Savva, A.: JSDL SPMD Application Extension, Version 1.0 (draft 008) Copyright Open Grid Forum (2006, 2007). All Rights Reserved
5. Theimer, M., Smith, C.: An Extensible Job Submission Design May 5, 2006 Copyright (C) Global Grid Forum (2006). All rights reserved, Copyright (C) 2006 by Microsoft Corporation and Platform Computing Corporation All rights reserved
6. Bylec, K., Mueller, S., Pabis, M., Wojtysiak, M., Wolniewicz, P.: Parametric Jobs – Facilitation of Instrument Elements Usage In Grid Applications. In: Remote Instrumentation Services on the e-Infrastructure. Springer US (2011), http://dx.doi.org/10.1007/978-1-4419-5574-62, 978-1-4419-5573-9
7. Lukasik, P., Sysel, M.: An Intranet Grid Computing Tool for Optimizing Server Loads. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Modern Trends and Techniques in Computer Science. AISC, vol. 285, pp. 467–474. Springer, Heidelberg (2014)
8. Rodero, I., Guim, F., Corbalan, J., Labarta, J.: How the JSDL can exploit the parallelism? In: Sixth IEEE International Symposium on Cluster Computing and the Grid, CCGRID 2006, May 16-19, vol. 1, pp. 275–282 (2006), http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1630829&isnumber=34197, doi:10.1109/CCGRID.2006.55
9. Developer Guide and Reference, Novell ® PlateSpin Orchestrate 2.0.2 (July 9, 2009), https://www.netiq.com/documentation/pso_orchestrate25/