Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Simplified implementation of medical image processing algorithms into a grid using a workflow management system[☆]

Dagmar Krefting[a,*], Michal Vossberg[a], Andreas Hoheisel[b], Thomas Tolxdorff[a]

[a] Institute of Medical Informatics, Charité - Universitätsmedizin Berlin, Germany
[b] Fraunhofer Institute for Computer Architecture and Software Technology, Berlin, Germany

ARTICLE INFO

Article history:
Received 28 November 2008
Received in revised form
16 April 2009
Accepted 22 July 2009
Available online 28 July 2009

Keywords:
Workflow management
Healthgrids
Image processing software

ABSTRACT

In this paper, we describe the grid integration of medical image processing applications as grid workflows. The workflow management system is able to execute all tasks related to grid communication, such as authorization, scheduling and monitoring. It remains to the developer to make the code accessible for the workflow manager, and to define, what to do with it. Coarse-grained parallelization of processing steps for runtime reduction can easily be realized. We describe the procedure how to port the code to the grid and show exemplarily the integration of segmentation and registration algorithms for transrectal ultrasound guided prostate biopsies.

1. Introduction

Many scenarios in medical image processing would benefit strongly from grids, may it be runtime reduction, collaboration by data and application sharing, or increased availability of algorithms independent of the user's location and local hard and software configurations [11]. However, if the expected effort for the grid integration is estimated to outbalance the (near-term) benefit, interested researchers are discouraged from using grid resources. The method proposed in this paper allows grid utilization without deep knowledge of the underlying infrastructure. Quick results can be achieved by basic implementation using XML-based workflow descriptions. The implementation can later be enhanced regarding user friendliness by integration into a grid portal.

2. Methods

The code is integrated into a production grid, based on *Globus Toolkit 4* (GTK4)[1]; and offering a grid-enabled *Picture Archiving System* (PACS) and the *Storage Resource Broker* (SRB) as data repositories [2,3]. For further details, please refer to [4]. The workflows are created by using the *Grid Workflow Description Language* (GWorkflowDL), which is an XML-based standard for describing workflows as *High Level Petri Nets* (HLPN) in accordance with the standard ISO/IEC 15909-1 [5,6]. Data is modeled as *tokens*, located at a *place*, and program execution is represented as a *transition*.

High level Petri nets can do anything that can be defined in terms of an algorithm and can be directly used to model transfer, execution and storage of any kind of input and output data as well as control data (e.g. the exit status of a workflow step). Flow control of the data through user constraints to the processor is a benefit over other data-flow languages like Scufl/XScufl, which are used by MOTEUR, a comparable workflow system for glite-based grid infrastructures [7,8]. The created workflow descriptions (WFD) are stored in an XML resource database, together with the software resource descriptions (SRD) and hardware resource descriptions (HRD). The latter utilize the *D-Grid Resource Description Language* (D-GRDL) and provide the information about available software and hardware resources within the grid. A WFD is then enacted by the *Grid Workflow Execution Service* (GWES), a workflow manager specifically targeted for WSRF-standardized grid usage [12]. The GWES utilizes GTK4's built-in security mechanisms (GSI), data transfer (gridftp) and generic job submission (WS-GRAM). WFDs can be realized on several abstraction levels, which are then concretized during runtime. The GWES provides basic resource brokering and meta-scheduling based on the current SRDs and HRDs. It also offers error handling strategies for fault tolerant execution. If an execution step fails, the error is reported and the transition is rescheduled to another resource up to an adjustable

☆ Funding from the German Federal Ministry of Education and Science, MediGRID (01AK803A-H) is gratefully acknowledged.

* Corresponding address: Institute of Medical Informatics, Charité - Universitätsmedizin Berlin, Hindenburgdamm 30, 12200 Berlin, Germany. Tel.: +49 30 450 544 515.
E-mail address: dagmar.krefting@charite.de (D. Krefting).
URL: http://www.medigrid.de (D. Krefting).

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.07.004

number of retrials. A generic GWES web user interface (GWUI) allows to upload workflow descriptions to the database, initiation and monitoring of running workflows. The different steps, that have to be processed by the developer to port an existing image processing program to the grid, are straightforward:

(i) *Software deployment to the gridnodes*

All software related to the specific application is stored at a separate directory on the connected clusters. The nodes of the clusters themselves are of homogeneous architecture, therefore it is sufficient to provide the appropriately compiled code on the respective head nodes. Developers can log on to these machines via *gsissh* and manage their application directories.

(ii) *Generation of a wrapper script*

During workflow execution, the GWES calls the software programs defined by the transitions; and the connected input tokens specify the input parameters. As the tokens in the net adhere in no particular order, GWES passes the parameters as key/value pairs and uses the name of the place as the key (i.e. *program-name -place1 token1 -place2 token2 …*). Since most existing programs do not conform to this plugin convention, a small wrapper script is required which is called instead of the actual program. In the wrapper script the parameter values are extracted from the key/value pairs and mapped to the program call. The wrapper script is also stored with the software on the head nodes and may contain auxiliary commands or environment settings.

(iii) *Registration of the software*

To publish the new software to the grid, a SRD file is generated and uploaded to the database. The SRD defines the URI of the software and contains the path of the respective wrapper script on the gridnode. An XML-element, containing the URI of the software, is added to all HRD files, where the software is available.

(iv) *Creation of a workflow description*

The workflow description itself consists of (a) a header providing the properties and execution information for the GWES, (b) place elements defining the places and initial assignment with tokens, and (c) transition elements defining the software to be executed and it's input and output places. Examples are publicly available on the *myExperiment* workflow repository.[1] We want to emphasize, that coarse-grained parallelization can be achieved by assigning multiple tokens, modeling different datasets, image slices or parameter sets, on a place. The individual tokens are then processed multi-threaded. The workflow description can then be uploaded and started using the GWUI. The actual state of the places (occupying tokens) and transitions (available and chosen hardware resource, state) as well as possible error messages or warnings are monitored. The upload of the workflow description is the default way for development and testing. The disadvantage is, that the datasets to be processed, i.e. the initial tokens, have to be inserted explicitly within the workflow description.

(v) *Optional: Integration into a grid portal*

More user friendliness is achieved by development of dedicated grid portlets, that are integrated into a Gridsphere portal.[2] The workflow descriptions are then provided as *string templates* within the portal, where they are completed by the initial tokens and are submitted by employment of the respective java packages.[3]

1 http://www.myexperiment.org/.
2 http://www.gridsphere.org.
3 *org.antlr.stringtemplate* and *net.kwfgrid.gwes*.

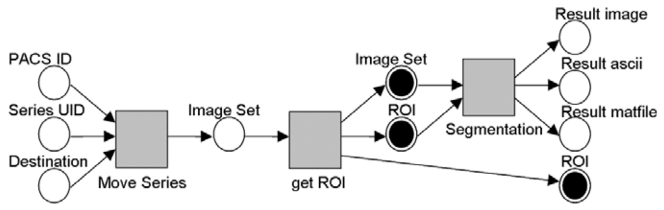

Fig. 1. Petri net of the segmentation workflow.

3. Results

To date, eight completely different applications have been successfully integrated to the grid with the described method, such as parallelized preprocessing of functional MRI data, hemodynamic simulations, gene prediction or biosignal analysis of polysomnographies. Each application may consist of one or more predefined workflow templates. Here, we present exemplarily the integration of algorithms to be used for analysis of transrectal ultrasound (TRUS) images taken during prostate biopsies. Prostate cancer is the most common cancer in men. Current goldstandard for prostate cancer diagnosis is a prostate biopsy, where ten tissue probes are taken from different parts of the prostate, guided by 2D transrectal ultrasound (TRUS) [9]. These TRUS images can now be utilized for automated determination of the location of the extracted tissue probes within the prostate volume, giving valuable information for diagnosis and therapy planning [10]. The main image processing steps are (a) localization of the extracted tissue probe within the 2D image by automated detection (segmentation) of the biopsy needle, and (b) registration of the 2D images into a previously taken 3D ultrasound volume. The two steps are mainly independent and can be processed in parallel; the segmentation and registration results are then combined in a final last processing step. For the sake of clarity, the last step is excluded here, and the two processing tasks are discussed separately. The needle segmentation consists of the following steps:

- Image transfer from the grid-enabled PACS.
- Calculation of the Region of Interest (ROI) on the first image of the series.
- Segmentation of the image series.

The workflow layout is given in Fig. 1. The code is written in *Matlab* and is compiled using Matlab's *Compiler Toolbox*.[4] Here, the main benefit from grid integration is the availability of the algorithm as a service for all grid users, independent of personal *Matlab* licenses. Some speed-up is achieved by parallel processing of the ten images from the different prostate locations, but has not been the main motivation. Runtime reduction is more important for the registration task. Here, the algorithms are developed in c/c++ using an intensity-based approach provided by the *Insight Segmentation and Registration Toolkit* (ITK).[5] Such an approach can be considered as a nonlinear optimization problem, where a certain similarity measure is optimized in the parameter space of translation and rotation of the 2D image relative within the 3D volume. Registration of the TRUS images have turned out to be a difficult task, as many local minima exist in the available similarity measures. Furthermore, the optimization algorithms implemented in ITK show poor performance in optimization of rotation angles. Therefore, a parameter scan with varying initial rotation angles is performed for each 2D image, from which the "global" minimum can be determined in a subsequent step. A single registration run needs in average 20 min. As an example, our local workstation

4 http://www.mathworks.com.
5 http://www.itk.org.