Dynamic GT4 Service Architecture to Distribute Scientific Software Components

Rainer Stotzka, Alexander Frank, Thomas Jejkal, Volker Hartmann, Michael Sutter, Hartmut Gemmeke

Forschungszentrum Karlsruhe,
Institute for Data Processing and Electronics,
Karlsruhe, Germany
rainer.stotzka@ipe.fzk.de

I. Objectives

The aim is to provide a dynamic Grid service architecture which can be used by scientists and extended by scientific software developers without prior knowledge of Grid technologies. With the use of WSRF–compliant web services based on the Globus Toolkit 4, software components are executed immediately allowing nearly real time handling of spontaneous requests.

II. Architecture

The Globus Toolkit 4 [1] supplies the key functionalities of the service architecture, e.g. the GT4 MDS Index Server and the Java WS core. The GT4 containers at the GT4 Worker Nodes are extended by the Marburg Hot Deployment Service (HDS) allowing to start and control services on a running GT4 container, and a Performance Measurement Service (PMS) to distribute scheduling information, e.g. the actual load. Security,

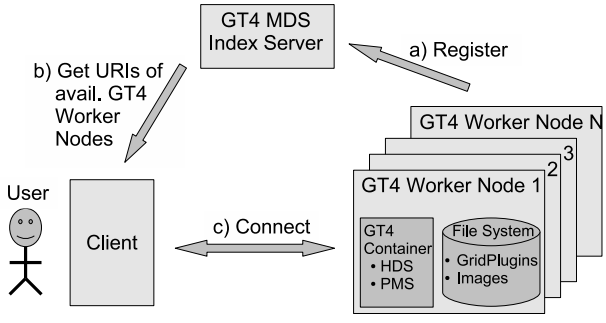

Fig. 1. The architecture consists of three parts: The GT4 Worker Nodes (right), the GT4 MDS Index Server (top) and the Client (left). a) When started, the GT4 Worker Nodes register at GT4 MDS Index Server. b) The Client, invoked by the user, inquires the MDS Index Server and gets the URIs of all available GT4 worker nodes. c) The Client connects to the GT4 Worker Nodes, requests their individual performance measurements, deploys services and interacts with them.

e.g. authentication, authorization and secure access to the deployed web services and data, is provided by GT4. The architecture is easily scalable and can be expanded by simply deploying two additional web services to GT4 computers or by using GT4Tray [2] to integrate desktop computers running Linux or Windows.

III. GridIJ

GridIJ is a reference implementation of the service architecture with a problem solving environment for image processing. It consist of:

- The Client software controlling an execution workflow,
- a graphical user interface,
- the GridPlugins, which are the software components written by scientific programmers using the GridPlugin interface and
- the IJPlugin Service managing the GridPlugins on the GT4 Worker Nodes.

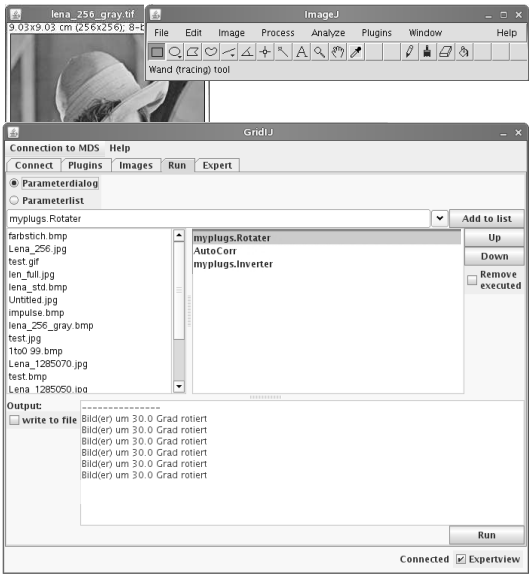

Fig. 2. ImageJ with GridIJ. Shown is the ImageJ graphical user interface (upper right window), an image (upper left) and the GridIJ graphical user interface (lower window). Within the active ”Run” tab of GridIJ the workflow of the deployed GridPlugins (right list) and the distributed images (left list) are present.

IV. Results

To give an estimation of the performance of the proposed GridIJ architecture an image registration algorithm based on local correlation has been implemented in GridIJ. The registration algorithm was originally realized as a stand–alone Java program using the ImageJ library. Modifying it took not more than 10 minutes programming time, demonstrating the ease of handling.

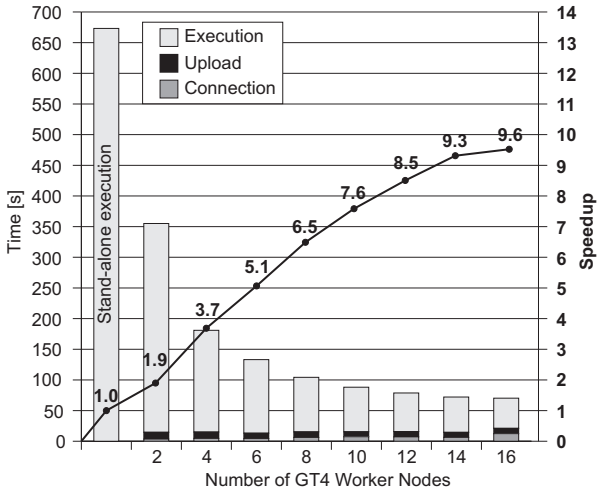

Fig. 3. Comparison of the execution, connection and upload times for the original program (stand–alone execution) and the corresponding GridPlugin with a growing number of GT4 Worker Nodes. The left axis shows the processing time represented in the plotted bars, the right axis (bold) shows the corresponding speedup represented in the line plot and bold numbers.

The performance of the GridIJ implementation with an increasing number of parallel GT4 Worker Nodes was compared to the original program run time (stand–alone execution) on a single workstation. Altogether the results show a typical behavior of parallel architectures: Figure 3 displays the processing time composed of execution, upload and connection times for different configurations, and their corresponding speedup. The workload for each GT4 Worker Node decreases, the more GT4 Worker Nodes are participating, and therefore the execution time decreases, like expected. The efficiency, defined as the ratio between the speedup and the number of GT4 Worker Nodes, can be regarded as a degree of the scalability. It decreases very within the observed range and arrives 0.6 for the maximum number of GT4 Worker Nodes. Since the connecting time and upload time remain similar, these administrative offsets will dominate the execution time for growing numbers of GT4 Worker Nodes resulting in an reduced efficiency.

V. Conclusion

The proposed design of a Grid computing architecture on the base of WSRF–compliant web services with dynamic service provisioning and management leads to an easy–to–use and an easy–to–extend Grid environment, which can be easily integrated in scientific problem solving environments. The scientific functions, embedded in web services, are executed immediately allowing nearly real time handling of spontaneous requests.

Due to the use of Java, WSRF–compliant Java web services and GT4Tray is is possible to build up highly heterogeneous architectures and integrating personal workstations running Windows as GT4 Worker Nodes.

References

[1] I. Foster, "Globus Toolkit version 4: Software for service-oriented systems," in *IFIP International Conference on Network and Parallel Computing*. Springer-Verlag LNCS 3779, 2005, pp. 2–13.

[2] T. Jejkal. (2006) GT4Tray website at Forschungszentrum Karlsruhe. [Online]. Available: http://fuzzy.fzk.de/~GRID/GT4Tray/