

Market-Based Grid Scheduling

In recent years, many papers and research projects have proposed and investigated the usage of economic principles (in particular auctions and negotiations) in allocating grid resources. The underlying idea is that markets perform well in allocating scarce resources among self-interested market participants. For instance, in grid settings, prices (i) cause the users to make efficient use of the resources and (ii) provide incentives to resource owners to contribute their idle resources to grids. However, only few of these projects have actually implemented economic logics into grid middleware, e.g. Globus Toolkit. One aim of the Biz2Grid project (<http://www.biz2grid.de>) is to bridge this apparent gap between theoretical research and practical implementation by integrating an economic framework into Globus Toolkit.

Conceptual Architecture of a Market-Based Grid System

The architecture – based on Globus Toolkit – aims at reducing the integration overhead by utilizing existing interfaces instead of building many additional interfaces on the upper layers. It consists of three layers, cf. Figure 1:

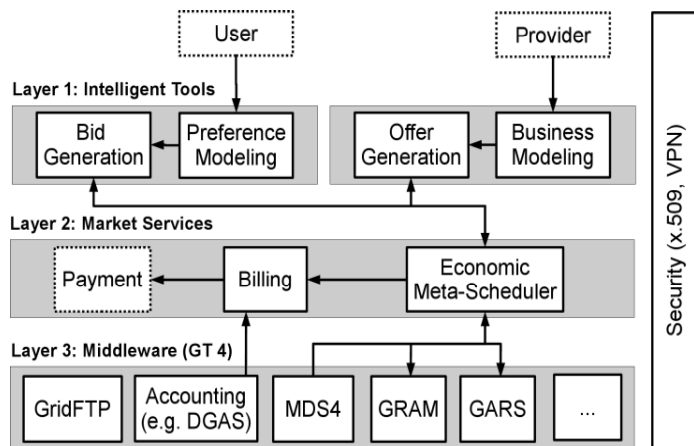


Figure 1. Conceptual architecture of a market-based grid system.

Layer 1 – Intelligent Tools: Participants (users and providers) cannot be expected to continuously monitor the market and interact with it. Consequently, configurable bidding agents are needed that automatically interact with the market on behalf of the participant, thus shielding parts of the system's complexity.

Layer 2 – Market Services: The economic meta-scheduler determines resource allocations and corresponding prices. Purely technical scheduling algorithms are enriched to take into account the participants' economic preferences.

From the user side, the technical attributes are retrieved from the job descriptions (e.g. JSDL), while the economic job attributes are obtained directly from the users' bidding agents. With respect to resource providers, the technical resource attributes are retrieved from the resource monitoring system, while the economic parameters are fed into the scheduler by the providers' bidding agents.

Layer 3 – Middleware: Besides the components mentioned above, the grid middleware contains a whole range of components that are not directly connected to the economic logic in the upper layers. For instance, the GridFTP component is used for the file transfer across the various clusters in the grid.

Integration into State-of-the-Art Middleware

A high-level overview of this implementation is depicted in Figure 2.

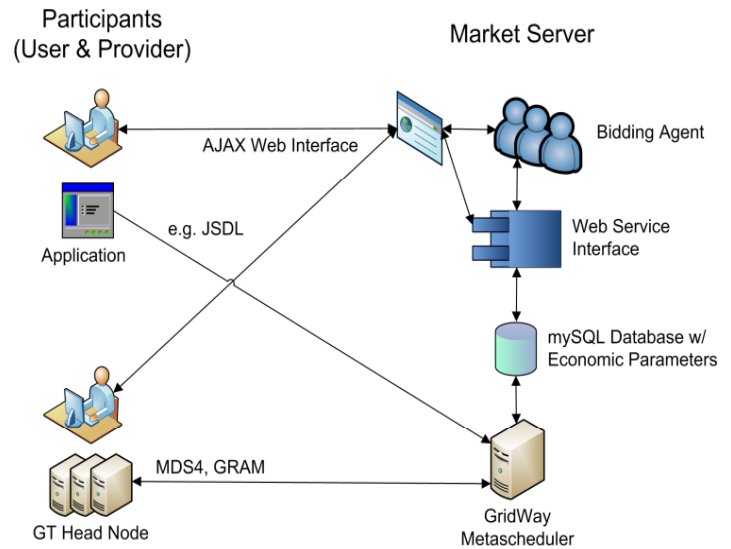


Figure 2. Instantiation of the conceptual architecture based on Globus Toolkit and GridWay.

The central part of our prototype is an SQL database that contains information about all jobs, head nodes (clusters), and participants. Users submit their jobs via the standard Globus Toolkit interfaces, e.g. by means of a JSDL file. Basic information about these jobs is written to the SQL database. The user can configure her bidding agent by means of a Web interface. This interface was built using the Google Web Toolkit. Since the Web interface is linked to this database via a Web service interface, the system can quite easily be configured to support a client-sided approach as well, thus bypassing the Web interface. The bidding agents are managed by a server thread that periodically checks the job table in the database for new entries.

The economic meta-scheduler has been implemented as a custom scheduling algorithm in the GridWay meta-scheduler. GridWay retrieves the technical job and node parameters from the JSDL file and the MDS4 component, respectively. The economic parameters are retrieved from the database. The outcome of this economic scheduling algorithm (allocations and prices) is then written back into the database and enforced by GridWay.

The main benefits of this prototype are that, by "piggybacking" on the GridWay meta-scheduler and the standard grid interfaces, the integration with the underlying grid middleware comes almost for free. All economic information is passed to GridWay via the SQL database, which results in a loose and modular coupling of the economic layers and the middleware layer. In consequence, the system can easily be configured to work without any of the economic components, but these essentially become optional and lightweight build-ins.

Project partners:

