Business Grid Components for WebSphere Extended Deployment – A Whitepaper

November, 2004

Overview

This paper provides a brief introduction to the **Business Grid Components for WebSphere®** **Extended Deployment** (Business Grid Components) – what they are, what kind of problems they solve, and what benefits they bring to an overall solution.

Background and Motivation

High performance computing (HPC) is a branch of computing concerned with applications that require extreme amounts of computational resource. Often found in the scientific, engineering and financial sectors, these applications consume large amounts of CPU, data storage or I/O (or combinations thereof). While the marketplace has addressed the needs of these applications in various ways, the current trend is toward grid computing where numerous, often modest machines are harnessed together to solve otherwise intractable problems. A grid scheduler is used to manage this collection of machines, determine how to best distribute the application workload among the resources it controls and effect those decisions. These grid schedulers have a different heritage and different characteristics than more traditional business scheduling products. While the two share many common traits, business schedulers tend to emphasize planning, calendaring and workflow while grid schedulers excel in resource monitoring and allocation, and more complex scheduling algorithms. The Business Grid Components focus on HPC/grid computing.

As hardware has become more powerful and grid schedulers have improved, customers have begun considering grid-based solutions for a wider range of applications. The same parallelization techniques that allow an application to achieve greater performance on a grid infrastructure can also improve its scalability and availability. In some cases these parallelization techniques can allow interactive use of applications that were previously limited to batch-only scenarios.

This has left customers with a dichotomy between their grid infrastructures and the infrastructure used to process transactional workloads (traditional Web applications, for example). Even though the two infrastructures are often built on roughly equivalent classes of hardware, they are provisioned from distinct pools of physical resources. Additionally they have distinct programming models, administration and management tools and are typically purchased from separate vendors.

Since customers over-provision their transactional infrastructure to handle peak workloads, there is often a substantial amount of unused computing power (“whitespace”) available on the transactional infrastructure. Operating efficiencies could be realized if we could interleave the transactional and grid workloads to more fully utilize the available resources. Such an environment would have to support policy-driven prioritization to ensure that the allocation of resources to the two types of workloads was consistent with the customer’s business objectives.

The Server Allocation for WebSphere Application Server project provided an initial proof-of-concept that such integration was feasible. It demonstrated that a WebSphere Application Server cluster could be used as a framework for distributing and performing the kind of computationally-intensive work typically done in grid environments.

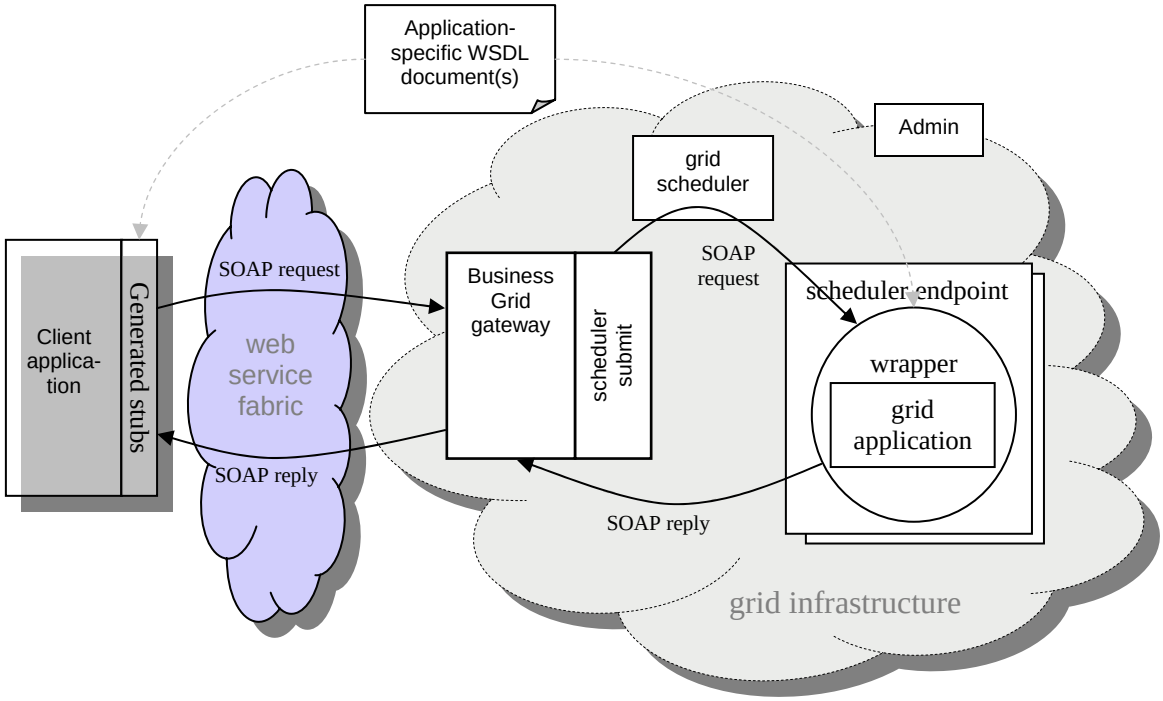
WebSphere Extended Deployment takes a different approach to optimizing resource utilization. For J2EE applications, it dynamically adapts the number and placement of application instances to reflect both the current load on the application and the relative policy-specified priority of the application. By doing this, the peak workloads of one application can be matched with valleys in the workload of another application and equivalent levels of service can be achieved with fewer total resources.

Business Grid Components, "A Cappella"

By themselves, the Business Grid Components provide a grid scheduling environment. Unlike many traditional grid schedulers, clients interact with the Business Grid Components using service-based mechanisms. Requests for service are expressed as Web service requests that conform to application-specific WSDL documents. Control functions (job query and cancel, for example) are exposed as Web services that conform to Business Grid WSDL documents. By exposing its grid functionality using SOA-based mechanisms, the Business Grid Components allow customers to leverage the significant development- and run-time tooling available for Web services and to extend their investments in Web service skills and infrastructure to include grid applications.

The application-specific WSDL documents are a key element of the Business Grid Components. Formally defining interfaces between pieces of the application allows the Business Grid Components to provide additional value to the application much like formally-defined interfaces allow the WebSphere Application Server to provide extra value to J2EE applications.

The following diagram shows a high-level component view of the Business Grid Components:



Business Grid Components

The Business Grid architecture can be described as five key components: the client application, the gateway, the scheduler, the scheduler endpoint/grid application and administration. Each of these is considered in its own section below along with an architectural description of each.

Client Application

The Business Grid Components do not restrict the runtime environment used by client applications. The architecture admits any runtime environment capable of initiating a request as described below. J2EE, J2SE, .Net, Perl and C/C++ are all possible languages/runtime environments for the client application. Similarly, the Business Grid architecture does not assume any particular impetus for these requests – they might be generated to handle an interactive user request, as a part of executing a workflow, or by a calendaring function, for example.

As previously mentioned, client applications interact with the Business Grid Components using a Web services programming model. Like most service oriented architectures, the Business Grid architecture prescribes the existence of one or more WSDL documents to describe the interactions between the client application and the Business Grid Components. These WSDL documents are defined by the application developer and are application-specific. That is, the operations described in the WSDL document are expressed in application terms and are not required to contain Business Grid-specific constructs. The Business Grid Components understand service requests expressed as SOAP over HTTP and SOAP over JMS.
Placing a service boundary between the client application and the Business Grid Components allows the client application to be written without hard-coded knowledge of how the requested service will be provided at runtime. Customers will be able to exploit future enhancements to the Business Grid Components without needing to update their client applications. It also allows the Business Grid Components and other Web-service-aware infrastructure to intercept, understand and modify the request at runtime to provide additional qualities of service.

Since the grid applications that will eventually provide the services described by these WSDL documents are resource intensive, it is likely the service requests will take minutes or even hours to complete. The synchronous exchanges usually associated with Web service requests are inappropriate for these long-running requests; rather, an asynchronous programming model is needed. Since there are currently no well-established asynchronous Web service programming models, the Business Grid Components provide various mechanisms to achieve asynchronicity:

1. Response service. The Business Grid Components provide tooling that decomposes request/response operations in a WSDL document into separate WSDL documents containing distinct one-way request and one-way response operations. The client application provides an implementation of the response service and specifies its location on the service request. The generated WSDL documents include additional WS-Addressing information to allow the application to specify both the location of the response service and a correlator to match the response with the request. The generated WSDL documents are then used by the client application developer in lieu of the original WSDL documents

2. Polling. The Business Grid Components permit the client application to poll the gateway to determine the completion status of outstanding service requests. The Business Grid Components provides tooling to modify the original WSDL document to include additional WS-Addressing correlator information to match the polling messages with the original request. Note that the use of polling limits the scalability of the gateway and should only be considered for client applications that cannot support any kind of callback.

3. Asynchronous JMS. The Business Grid Components provide additional tooling and runtime infrastructure to support an asynchronous JMS interface to the Web services.

Regardless of which mechanism is selected, the client application developer uses standard tooling to generate Web service stubs from the modified WSDL documents.

Gateway

The Business Grid gateway maps between Web service requests from client applications and the job scheduling functionality provided by the scheduler.

To allow the gateway to handle requests for arbitrary applications, the gateway code itself is ignorant of the semantics of any particular application. When an application is deployed using the Business Grid Components, an application-specific transform is deployed to the gateway instances. This transform specifies how to change an incoming SOAP service request to a job specification the scheduler can interpret.

In addition to application-specific service requests, the gateway also supports job control requests to query, cancel and get the results of previously submitted service requests.

Scheduler

Like any other grid scheduler, the scheduler component of the Business Grid Components is responsible for classifying, prioritizing and executing the requested jobs according to a set of customer-defined policies. From a functional point of view, it is the workhorse of the Business Grid Components function and provides the most visible value to the customer. The Business Grid Components include an integrated, fully functional scheduler. Since the scheduler is key to the overall functionality of the Business Grid Components, improving this integrated scheduler will be an ongoing priority.

There is a basic set of grid scheduling functions needed to support the kinds of applications described later in the “Application Patterns” section. The scheduler used by the Business Grid Components support the following functions/features:

- schedule both serial and parallel (MPI) jobs
- submit, query and cancel jobs
- prioritize jobs according to customer-defined criteria
- dynamically add and remove machines from the pool of machines available to execute jobs
- programmatic or command-line interfaces to functionality

In addition to these basic requirements, the integrated scheduler provided with the Business Grid Component provides these additional functions/features:

- ability to choose among several scheduling algorithms, including backfill scheduling
- support for multi-step jobs with conditional execution
- fault tolerant scheduler infrastructure
- customer-defined resources and floating resources (software licenses, for example) in addition to machine-specific resources (memory, CPU, etc.)
- preemption, checkpoint and restart facilities
- graphical, command-line and programmatic interfaces to all functionality
- collection of job accounting data
- OS-assisted enforcement of job resource limits

See the “First Steps” section for more details on the scheduling technology included in Business Grid Components.

Scheduler Endpoint/Grid Application

The grid application running on the scheduler endpoint is responsible for the actual execution of the service requested by the client application and transmission of the result back to the gateway.

As with the client application, the Business Grid Components use a Web service interface to communicate with the grid application. Introducing a service boundary between the scheduler and the grid application allows the grid application to be built using standardized tooling and to be scheduler-agnostic.

Due to the resource demands of the applications themselves and the nature of current grid schedulers, the grid application that runs on the endpoint is often, but not always, written in a language like C or C++ and compiled for a specific combination of hardware and operating system. As long as the application can be run by the scheduler, however, the Business Grid Components do not place restrictions on the exact runtime environment used by the grid application (standalone native executable, shared library, Java™ classes, etc.).

The grid application developer builds one or more wrappers that, for each operation described in the WSDL documents, accept a SOAP request conforming to the WSDL document, invoke the application to perform the requested action, then generate a SOAP response conforming to the WSDL document. The Business Grid Components must be able to invoke these wrappers, but do not place any additional restrictions on how they are constructed or the runtime environment in which they reside. Shell scripts using libxslt, gSOAP, Apache Axis C++, Perl and Python are among the dozens of possible runtime environments for the wrapper. If the application itself meets the SOAP processing requirements described above or can be modified to do so (for example, by integrating it with gSOAP), a separate wrapper might not be required at all. The Business Grid Components will include sample applications that demonstrate how to use a small number of key environments.

Administration

Underlying and controlling the pieces described above is a rich set of administration functions. The Business Grid Components allow an administrator to define policies and service levels to govern how work requested by client applications is scheduled. Additional data collection and visualization tools show administrators how the system's resources have been allocated and how well the system is achieving the goals they have put in place. Finally, application deployment mechanisms allow administrators to install grid applications centrally and have them transparently installed on the scheduler endpoints where they will eventually run.

Benefits

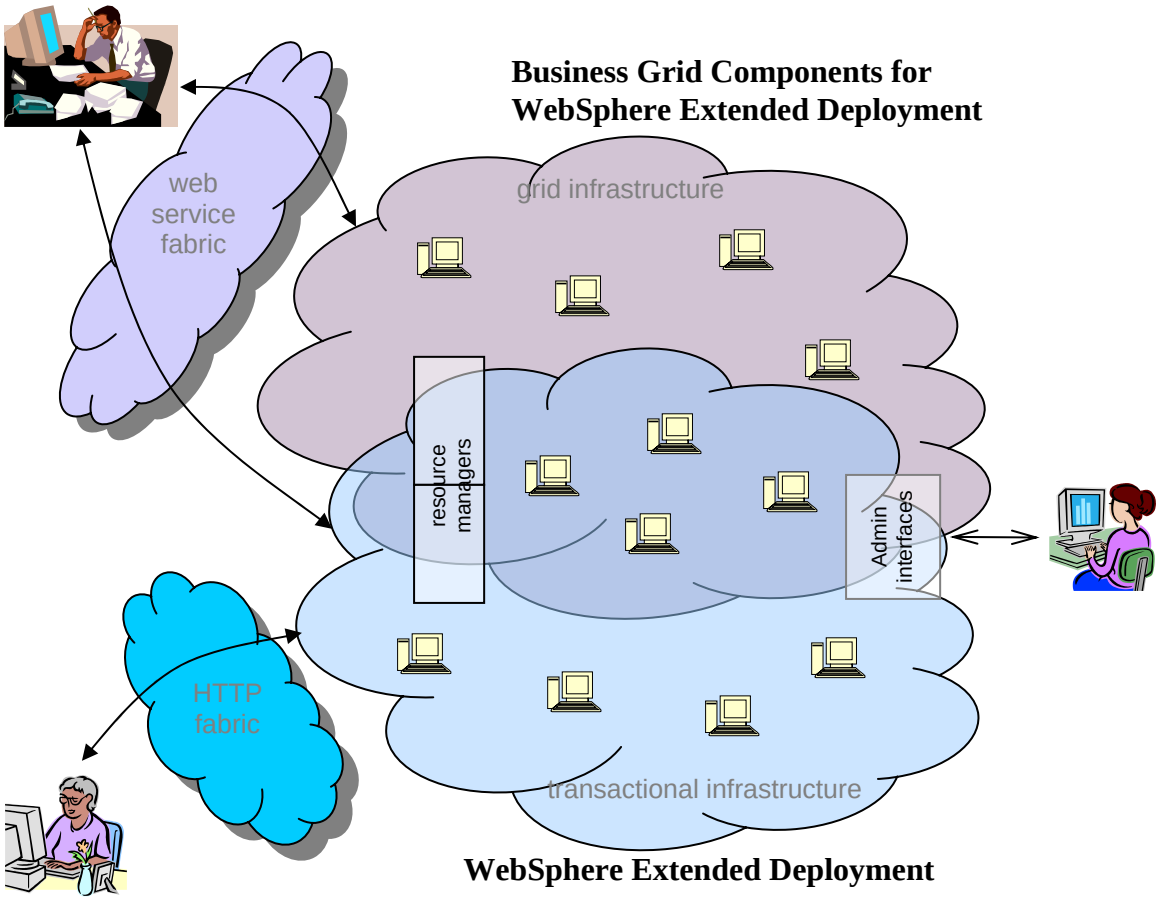
The Business Grid Components allow application developers to express their HPC/grid requests in application-specific terms rather than a scheduler-specific job language. This clean, SOA-based separation of "what to do" from "how to do it" has a number of benefits:

- It insulates the client application developer from specifics of the job scheduling backend, allowing them to focus on the client application rather than building skills in scheduler-specific job languages.
- The scheduler on the backend can be swapped with no impact to the client application. This allows the customer to upgrade to better scheduling technologies as they become available while still preserving their client application code investments.
- It brings many of the benefits of a managed application environment to HPC/grid applications. Aspects of the application such as qualities of service can be specified by administrators at deployment rather than being hard-coded in application artifacts.
- It allows a common programming model (Web services) to be used for both transactional and HPC/grid requests. Application developers can leverage a single set of skills and tools for both types of requests.

- As technologies such as the enterprise service bus evolve, they can be applied to HPC/grid-based applications without requiring application changes.

Business Grid Components with WebSphere Extended Deployment

When the Business Grid Components are combined with WebSphere Extended Deployment, however, even more operational efficiencies and potential cost savings are realized. The Business Grid Components integrate with WebSphere Extended Deployment to create an on demand infrastructure that spans both grid and transactional workloads. This integrated environment autonomically determines how to optimally allocate a pool of resources among transactional and grid workloads to best achieve customer-defined goals. By supporting both transactional and grid infrastructures, WebSphere Extended Deployment and the Business Grid Components allow more efficient resource utilization and better integration of administration, visualization and policy functions.



WebSphere Extended Deployment introduces the notion of dynamic application placement – that is, instances of a J2EE application can be started and stopped as warranted by current demand for the application and administratively-defined policy. Integrating WebSphere Extended Deployment with the Business Grid Components extends this concept to additionally include grid applications.

WebSphere Extended Deployment and the Business Grid Components cooperate to determine how the available resources can best be used to achieve the goals defined to them by an administrator. Resources are shifted between the transactional and grid workloads as warranted

by the current workload, in accordance with customer-defined goals. Like WebSphere Extended Deployment, an integrated WebSphere Extended Deployment and Business Grid environment works with IBM Tivoli® Intelligent Orchestrator to support dynamic provisioning scenarios that extend beyond WebSphere Extended Deployment and the Business Grid.

In addition to integrated resource management, WebSphere Extended Deployment and the Business Grid Components also support integrated administration, policy definition and reporting/visualization.

Benefits
Integrating the transactional and HPC/grid infrastructures using WebSphere Extended Deployment and the Business Grid Components provides a number of benefits:

- It allows consolidation of the IT resources used for transactional and HPC/grid workloads. It reduces the cost of over-provisioning by allowing the over-provisioning to be done once on a shared set of resources rather than twice for independent sets of resources.
- Integration of the two infrastructures allows more efficient utilization of resources. By having greater flexibility in the types of work that can be scheduled, the Business Grid Components is better able to maximize the customer's IT investments.
- It allows us to extend the value propositions of WebSphere Extended Deployment and the on demand Operating Environment to HPC/grid workloads. HPC/grid schedulers have long had the ability to specify policy for controlling resource allocation. The Business Grid Components augment this with more dynamic mechanisms such as SLA metrics and dynamic provisioning.

Application Patterns
The "grid application" label applies to a very broad range of applications based on a large number of application patterns. The Business Grid Components support the following subset of grid application patterns.

1. Embarrassingly Parallel. These applications can be easily divided into parallel units of work that do not require communication either among themselves or with the code that initiated the work. In the embarrassingly parallel pattern, a client application requests some number of instances of work to be run in parallel and assumes responsibility for collecting and aggregating the various results. The work is expressed as multiple service requests from the client, each of which results in the execution of a single instance of the grid application.

2. Parallel grid application. In this pattern, it is the grid application itself rather than the client that manages the parallelism. The client application issues a single service request that triggers a parallel execution of the grid application. The amount of parallelism may either be specified by the client application or hardcoded in the grid application. The grid application expects the parallel units of work to run concurrently and typically uses libraries such as MPI to allow communication among the distributed pieces. The grid application is responsible for aggregating intermediate results and producing a single final result for the client application.

3. Stateful grid application. This style of application uses multiple, ongoing interactions between the client application and the grid application to allow the client application to guide the grid application's efforts. The running instance of the grid application contains state information needed to process the client application's requests. In some cases, the state information is required to correctly process the client application's requests; in other

cases, it simply allows the grid application to do so more efficiently. The client application typically assumes responsibility for terminating the grid application after the final work request has completed. The grid application itself may either be a single process or a parallel grid application as described above.

First Steps

As with any complex undertaking, it is not possible to deliver the entire vision described above in a single release. The initial pieces of Business Grid Components will be delivered as a technology preview with WebSphere Extended Deployment, Version 5.1. The main focus of this initial release is to establish the programming model for client and grid applications so that we can evolve the underlying scheduling technologies without impacting customer code.

In this release, Business Grid Components are only supported in conjunction with WebSphere Extended Deployment. The following sections describe the functionality in this initial technology preview.

Client Application

XSL transforms will be provided to generate the request/response, polling and asynchronous JMS WSDL documents from the original documents as described above. The XSL transforms use only standard features and should work with any XSLT 1.0 conformant transform engine (Apache Xalan, Ant, J2SE 1.4, xsltproc, etc.).

Gateway

The gateway function will be provided as an additional set of code deployed on top of WebSphere Extended Deployment. The gateway will accept initial service requests and subsequent control requests sent using SOAP over HTTP or SOAP over JMS. Query, cancel and get-result job control requests will be supported. The application-specific transform from a service request to LoadLeveler job (see below) must be expressed as an XSLT 1.0 transform.

Scheduler

The scheduling mechanism used in the initial Business Grid Components technology preview will be IBM LoadLeveler version 3.2 running on 32-bit Intel Linux machines. Query and cancel functionality will be mapped from the gateway to the corresponding LoadLeveler functions. In addition to embarrassingly parallel applications, LoadLeveler supports more sophisticated forms of parallelism including gang scheduling and MPI.

Endpoint Environment

A utility script will be provided on the endpoint to assist the grid application with handling the WS-Addressing information needed by the gateway to correlate requests and responses. This utility script invokes the grid application's wrapper script and supplies the SOAP request to the wrapper on its stdin stream. The utility script requires the wrapper to produce the SOAP response message on its stdout stream.

Administration

Tooling to move computational resources between transactional and grid workloads will be provided. A basic set of visualizations will allow the administrator to see how resources are being utilized for grid workloads. A rudimentary deployment mechanism will be provided to distribute the grid application to the LoadLeveler endpoints. Policies to control the scheduling of jobs must be specified using LoadLeveler mechanisms.

Miscellaneous
Two fully functional sample applications will be provided. These samples include client applications, request transforms for the gateway and endpoint code.

While much work remains to be done to achieve the overarching Business Grid Components vision, this initial release is an important first step. As a vehicle for engaging early adopters to start building pilot projects, it will provide valuable customer feedback that will guide refinements to the Business Grid architecture and implementation.

Work is already in progress on future enhancements to the Business Grid Components. The two primary areas of focus are (1) tighter integration between the transactional and grid resource managers to support more efficient resource allocation and better administrative tooling, and (2) improvements to the underlying grid scheduler.

Conclusion

The Business Grid Components for WebSphere Extended Deployment consolidate the programming model and runtime infrastructures associated with transactional and HPC/grid workloads. This consolidation allows the customer to reduce operating expenses now and prepares their environment to better leverage future improvements in SOA and on demand technologies.

Authors

Mike Burr (burrm@us.ibm.com)
Gennaro (Jerry) Cuomo (gcuomo@us.ibm.com)

For more information

To learn more about IBM WebSphere Extended Deployment, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/webservers/appserv/extend/

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
11-04