DELIVERABLE D2.2

PART III: DESIGN DOCUMENT FOR GRIDBENCH

Task 2.3 Benchmarks and Metrics

Document Filename:	**CG2.3-D2.2-v1.0-UCY005-GridBenchDesign**
Work package:	**WP2 Grid Application Programming Environment**
Partner(s):	**UCY**
Lead Partner:	**UCY**
Config ID:	**CG2.3-D2.2-v1.0-UCY005-GridBenchDesign**
Document classification:	**CONFIDENTIAL**

Abstract: This document describes the early design for GridBench, the CrossGrid Benchmark Suite. GridBench is comprised of three families of benchmarking codes: Micro-benchmarks, Micro-kernels and Application kernels. GridBench will be used to investigate experimentally the performance of the CrossGrid testbed. In particular, it will be used to measure the performance of different elements of the CrossGrid architecture. Performance measurement through benchmarking is expected to provide information that is useful for performance comparisons of different Grid configurations, for investigating the effect of Grid architecture and available resources to application performance, for Grid administration, performance prediction, cost estimation, etc.

Delivery Slip

	Name	Partner	Date	Signature
From	M. Dikaiakos, G. Tsouloupas	UCY	09.08.2002	
Verified by	Jorge Gomes, Mario David, João Martins	LIP	21.08.2002	
Approved by				

Document Log

Version	Date	Summary of changes	Author
0.9	9/8/02	Draft A	M. Dikaiakos, G. Tsouloupas
0.91	28/8/02	Pre-Final	M. Dikaiakos, G. Tsouloupas
1.0	30/8/02	Final	M. Dikaiakos, G. Tsouloupas

Table of Contents

1. EXECUTIVE SUMMARY

As described in the Technical Annex and SRS of T2.3, it is the purpose of Task 2.3 to provide "Benchmarks and Metrics". Task 2.3 proposes a suite of benchmarks, GridBench - the CrossGrid Benchmark Suite. This suite with the appropriate metrics, aims to capture concisely the performance of Grid constellations, and of Grid applications deployed upon a Grid constellation. The suite will be carefully selected and adapted from existing benchmark libraries. The main criteria for selection are:

1. Applicability/utility to grid environments,
2. availability of "paper-and-pencil" definitions and
3. simplicity.

In addition to the selected benchmarks, new ones will be defined in "paper-and-pencil" and then developed, to reflect CrossGrid Applications more closely. Reference implementations will be provided for all "paper-and-pencil" definitions that are either i) developed by Task 2.3 or ii) are selected but have not been implemented yet.

In order to implement and run the benchmark suite on the CrossGrid test-bed it is required that alongside the benchmark suite we provide a set of administrative utilities that function as user and system interfaces. More explicitly Task 2.3 will:

- Provide the tools and utilities (in cooperation with Task 3.1, Portals) that are necessary to enable users to execute benchmarks and view results in a useful and user-friendly way.
- Provide members of the CrossGrid consortium with access to the benchmark suite and to historical data collected from benchmark invocations, through the CrossGrid portal interface.

2. INTRODUCTION

A benchmark:

- Enables one to state the capabilities of a system in a way that another can appreciate (by comparison to a similar set of results from another system)
- Helps one assess the performance of different types of computation on different architectures
- Can serve as an aid to predicting performance/cost of running an application
- Can evaluate an architecture thus helping to improve it or help develop other architectures
- Can determine the peak performance of a system and compare it to the performance of a real application
- Helps one discover bottlenecks and problems

Generalizing, the abundance of benchmarks can be hugely attributed to the multitude of variables in a parallel system that affect performance. It is one of the main purposes of Task 2.3 to select and adapt from the existing abundance of benchmarks (as of course they apply to Grid environments).

Determining performance in a parallel system is difficult enough (mainly because of the multitude of factors that affect it). This is even more of a problem when it comes to determining performance of Grid environments because of the additional and more complex factors that come into play.

There are two main methodologies/approaches for measuring performance of a complex system; first is to measure each individual component on its own, and second is to try to measure all of them together (the system as a whole). Measuring one performance factor at a time can be accomplished by using micro-benchmarks, which by definition measure one aspect of the system at a time. The utility of micro-benchmarks is their ability to pinpoint problems and provide feedback for subsystem modifications. Micro-benchmarks can also play a role in determining whether a problem lies in the hardware, middleware or application. Still, even if we measure each performance factor accurately (and use the right metric for each factor) it tells us little about the performance of the system as a whole.

In the absence of a usable, detailed and powerful theoretical model that works with most real applications and architectures and which can encompass all these low level metrics of performance factors to produce an accurate performance estimate, one has to resort to kernel or application benchmarks. Micro-kernel benchmarks can measure the performance of the system as a whole to perform computations similar to those of the micro-benchmark. Kernel benchmarks are usually scaled down applications and while they are useful in that they perform common and interesting computations, they run the risk of making too many assumptions on which parts of an application are the most important [1]. On the other hand, micro-benchmarks are required to fully understand and assess the results of more high-level benchmarks. Application benchmarks and application-kernel benchmarks aim to take this one step further and try to measure the ability of a system to run applications that perform more than just computations of one type. An application benchmark on a parallel system could try to capture its ability to perform several diverse computations as well as other operations (storage and retrieval form a storage system for example), possibly concurrently.

Also challenging is the definition of performance itself, i.e. the metric or set of metrics that concisely describe the capabilities of the system. Decades of research efforts in the area of performance measurement have not come to a single universally accepted metric, though some are more accepted than others. What's more, applications vary extensively in structure, communication patterns, memory access patterns etc., so one can only speak of system performance in relation with a specific application, or application type.

In a grid environment an application benchmark can measure the systems ability to perform computations, large data transfers, deliver interactive content, store and retrieve large amounts of data. An application benchmark should be representative and shared [1]. Being representative, the benchmarks should reflect real application performance. This usually involves using an application that is well-known and understandable for users. The question then easily arises: Which application is the one that addresses all -or most- aspects of the system? Even if such an application exists - one that performs all of the above in a "balanced" way (which raises an issue of whether there can be a right balance) - how representative is it of other real applications?

2.1. PURPOSE

The first part of the introduction poses some challenges for measuring performance on grid environments. The purpose of the rest of this document is to describe a toolbox for determining the performance of Grid environments. The GridBench suite will provide tools to measure the performance of the Grid at different levels: i) the CE/SE level, ii) the Site level and iii) the Grid Level. To do this, three main methodologies will be employed: i) micro-benchmarks, ii) micro-kernel benchmarks and iii) application kernel benchmarks. For descriptions of the utility of measuring performance at different levels and employing different methodologies to do it see the SRS of Task 2.3.

2.2. DEFINITIONS, ABBREVIATIONS AND ACRONYMS

CE	Compute Element
APART	Automatic Performance Analysis: Real Tools
API	Application Program Interface
CG	CrossGrid
CrossGrid	The EU CrossGrid Project IST-2001-32243
DataGrid	The EU DataGrid Project IST-2000-25182
G_PM	Grid-enabled Performance Measurement tool
Grid Constellation	A collection of "sites" (see "*Site*"), connected within a Virtual Organization and used for the solution of a common problem
GridBench	The CrossGrid benchmark suite
GridFTP	A Grid-enabled version of FTP (Part of Globus)
HEP	High Energy Physics
HLAC	High-level analysis component
HPC	High Performance Computing
HTC	High Throughput Computing
LDAP	Lightweight Directory Access Protocol
MPI	Message Passing Interface
NAS	NASA Advanced Supercomputing
RDBMS	Relational DataBase Management System
SE	Storage Element
Site	A set of Computing Elements, Storage Elements or other resources in a single geographical location.
SRS	Software Requirements Specification
UML	Unified Modeling Language
VO	Virtual Organization
XML	eXtensible Markup Language

3. SYSTEM DECOMPOSITION DESCRIPTION

To meet the requirements detailed in the SRS of Task 2.3 two sets of applications will be developed.

- GridBench: the CrossGrid suite of benchmarks, and
- Administrative utilities for running benchmarks and interfaces with external components

Figure 1 shows the high level diagrammatic view of the system.

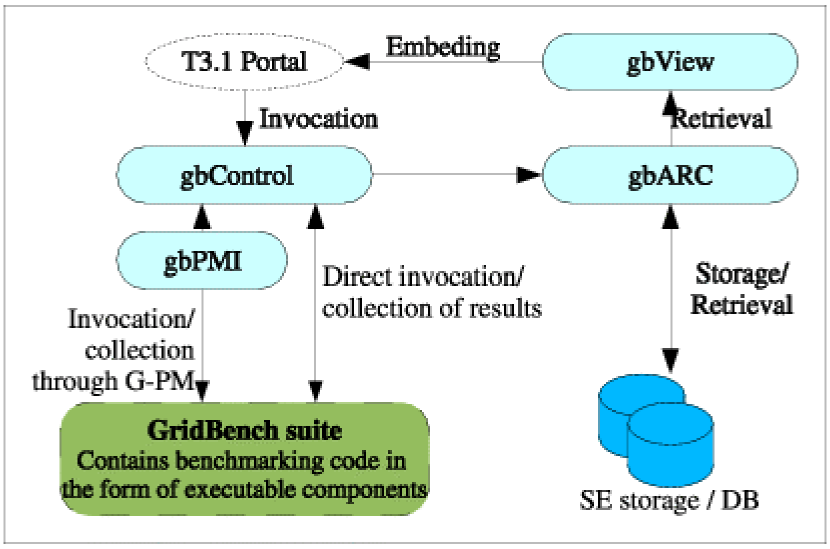
T3.1 Portal
Embeding
gbView
Invocation
Retrieval
gbControl
gbARC
gbPMI
Direct invocation/ collection of results
Storage/ Retrieval
Invocation/ collection through G-PM
GridBench suite
Contains benchmarking code in the form of executable components
SE storage / DB

Figure 1: A high-level diagrammatic view of GridBench.

3.1. GRIDBENCH

The set of components that make up the GridBench suite have the purpose of measuring the performance of the CrossGrid testbed and its constituents. To do so effectively, the suite will have to capture the performance at different levels. The "levels" of the Grid are i) the CE/SE (or machine) level, ii) the Site level and iii) the Grid level. Apart from the point of view of the Grid layers, GridBench will also provide different approaches to measuring performance: Micro-benchmarks, Micro-Kernels and Application kernels. (More on these can be found in [2].)

3.1.1. Benchmark module templates

While there is need for measurements at different levels with different methodologies, it is desirable to provide a coherent way of looking at the benchmarks - especially their results. While providing a coherent interface to all benchmarks in the suite may be difficult in some cases (simply because of the diverse nature of the benchmarks), uniformity will help both system integrators and users. What follows is an initial attempt to provide a common interface to benchmarks:

Each benchmark will be contained in one logical module, though some benchmarks will be made up of more than one module, which will conform to a common template. This interface will be implemented by all benchmarks. The data required for, and produced by each benchmark will fit into this template and also into a common XML schema for transport and storage.

The main requirements for the contents of such a template will be:

- Metadata
- Inputs
- Outputs

Metadata provide general, static information on the benchmark such as:

- An identifying name;
- A very short description;
- Parameter list with allowable values;
- General input requirements;
- Allowable targets (CE/SE/Cluster/Grid);
- Output metrics definitions.

Inputs provide data that is potentially unique for each invocation:

- Parameter values;
- Input files;

Execution specifications (time/location as requested by user).

Outputs provide singular, 1D or 2D value sets that represent measurements for each invocation:

- Metric values
- Output files (potentially inputs to other benchmark modules)

Execution specifications (time/location as delivered by system)

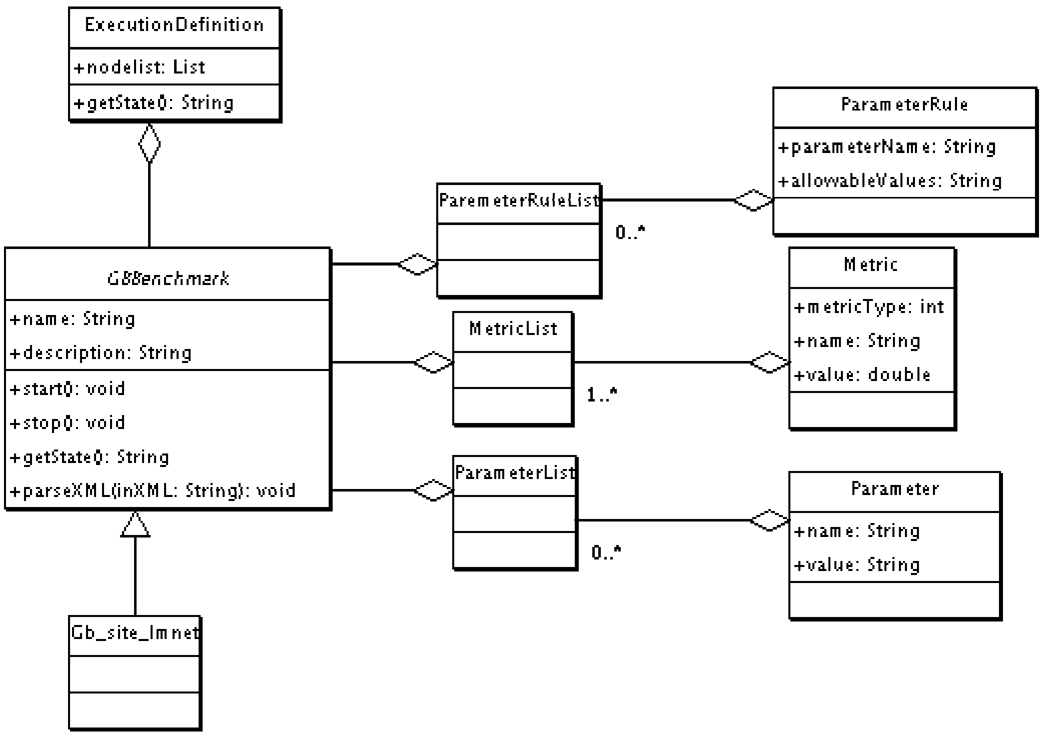

Figure 2: GridBench suite common benchmark interface

Error! Reference source not found. is a UML diagram that illustrates how the different modules of the GridBench suite derive from the common template. Class **Gb_site_lmnet** is a site-level micro-benchmark and it is included in **Error! Reference source not found.** to show that GridBench benchmarks will implement the GBBenchmark interface as in Figure 3.

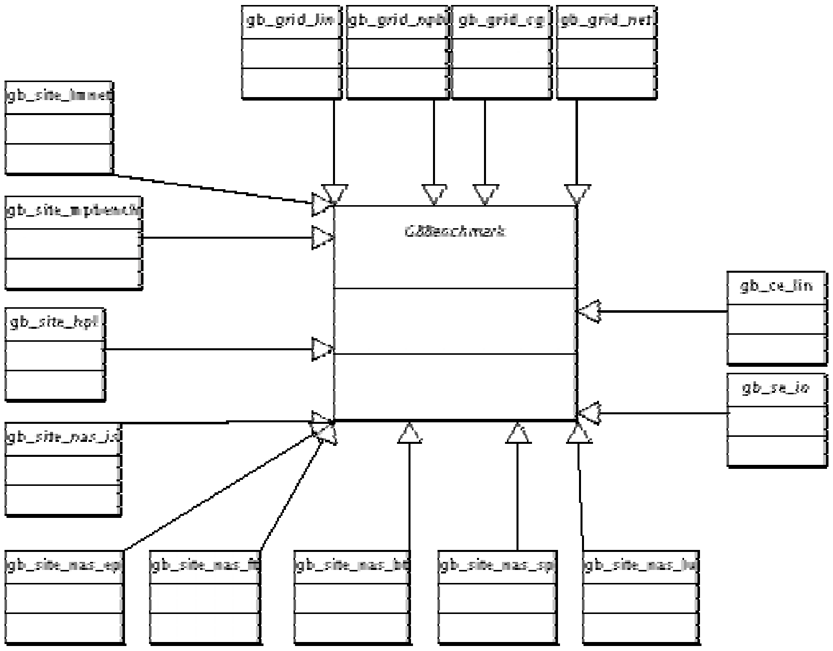

Figure 3: All GridBench benchmarks implement the GBBenchmark interface.

4. DEPENDENCY DESCRIPTION

4.1. MIDDLEWARE

Software produced by Task2.3 will assume full functionality of the Globus and other middleware as outlined in the SRS of Task2.3[2]. Whenever a user launches a particular benchmark, he will access performance measurements of relevance via the tools of WP2.4 (G-PM Performance Monitoring tool).

4.2. PORTALS

From the point of view of CrossGrid middleware, benchmarks are nothing more than simple applications that can be launched on the resources of CrossGrid via the CrossGrid portal. The basic difference between a benchmark and a real application is that the benchmark is an implementation of a well-defined program (in paper and pencil), with performance properties that are thoroughly analyzed and well understood.

Therefore, the user of CrossGrid benchmarks should be able to use the portal to login and define a benchmarking job in terms of the particular benchmark that he wants to execute (among a suite of supported benchmarks), its configuration (input files, configuration parameters), and the resources he wants to reserve for his job. Site administrators should also be able to define and run benchmarks, collect and store measurements in databases for the perusal of Grid users.

In contrast to real applications, which are executed to provide some useful output data, benchmarks are executed simply for collecting performance measurements. These measurements will be also summarized through the CrossGrid portal in hypertextual (HTML) and graphical format. In some cases, the actual benchmark output is of interest as it can be used as a proof of a benchmarks algorithmic correctness.

4.2.1. Interoperability with Task 3.1

This section describes the User Portal that provides access to GridBench, the CrossGrid benchmarks (task 2.3).

The GridBench suite enables a user to select benchmarks and perform benchmarking experiments.

The User Portal provides services:

- to give access to a database of prior benchmarking results;
- to define benchmarking jobs;

to request the job output in graphical and/or alphanumerical format.

Data files required as input for particular benchmarks are saved in storage elements and described in a METADATA catalogue encoded in XML format. This METADATA will be provided by the gbARC service.

The following is a list of actions that a GridBench user can perform in a general scenario:

A user starts with the login operation authenticating him using his certificate to be accepted as a member of the CrossGrid VO.

Once accepted, the user sees a list of available benchmarks and associated input files and parameters on the client.

Within this list, the user selects the relevant benchmark to be executed in the computational grid.

The user defines the input data for the selected benchmark. The user submits the selected GridBench job with the defined input data.

If the user feels the benchmark takes too long to execute, he can send a request to stop it (like CTRL-C command).

At the end of the session, the user can save the whole session trace in XML format either on their local storage or on the benchmark repository within the system for the perusal of other users.

4.2.2. User Interface

The graphical interface between the user and the GridBench will be provided in the context of the GRID desktop and implemented as a module of the Application Portal Server. A user will employ the various input panels of the Application Portal Server to login, select input files and benchmarks, and launch a benchmarking experiment. Aggregate benchmarking results will be presented through the Benchmark and Metrics Panel. Additionally, the user will be able to monitor measured performance properties through the user interface of the G-PM module of WP2.4. The UI will provide two options for reviewing performance measurements:

Through a hypertextual interface, implemented as an HTML page and enabling the user to click on the descriptions of machines/sites upon which a benchmark was executed in order to view the performance metrics derived from his benchmarking experiment.

Through a graphical representation of the high-level, abstract model of the Grid configuration used. The user will be able to click upon the icons of the machines/sites upon which the benchmark was executed in order to retrieve the performance metrics derived from his benchmarking experiment.

Figure 4 presents a high-level view of the graphical user-interface.

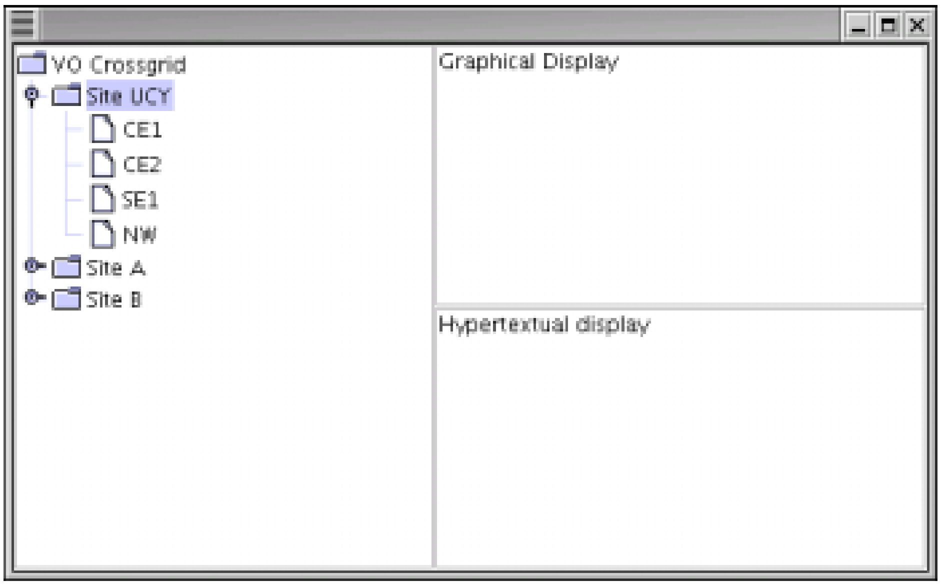

Figure 4: High-level view of graphical UI

4.2.3. System Issues

The User-Interface will be implemented with a GridBench Java servlet that will be integrated with the servlets of the Application Portal Server. The output of benchmarking experiments will be encoded in XML and stored in files and/or LDAP/RDBMS databases.

4.2.4. Assumptions and Dependencies

For the generation of the graphical representation of the benchmarking results, the UI servlet should have access to a file describing the various resources available through the CrossGrid platform, in order to generate the proper graphical representation. This file could be given in a common XML format.

4.3. GRID MONITORING

Task 2.4 plans to develop G-PM, a tool for providing performance measurement information. In particular, G-PM: (i) collects measurements of various aspects of program execution on the Grid, through its Performance Measurement Component (PMC); (ii) extracts high-level performance properties of an application through its High-Level Analysis Component (HLAC); (iii) predicts application performance based on an analytical modeling and prediction module, the Performance Prediction Component (PPC); and (iv) enables the visualization of performance data through its User Interface and Visualization Component (UIVC). Of particular importance for the benchmarking measurements is the employment of the HLAC component of G-PM. HLAC "allows to measure application specific performance properties" by "inserting probes at strategic places of the application code"[3]. HLAC employs the APART Specification Language ASL {Kesselman, 2000 #17} to specify, "how application specific properties are computed from the data delivered by these probes and the generic performance measurement data" [3]. Therefore, and according to what is expected from applications by the G-PM [3], the ASL specification of all GridBench specific performance properties will be stored in configuration files compliant to HLAC specifications. These files will drive the measurements and reports of GridBench metrics via the G-PM tools. Notably, the configuration files provided by GridBench will provide an automated alternative to the manual configuration of measurements through the "measurement definition window" of User Interface of the G-PM. Besides application-related requests, the G-PM tools enable the execution of infrastructure-related requests that specify "information to be returned on the whole or part of the Grid infrastructure," such as the availability of resources, static and dynamic resource information, etc [3]. Infrastructure-related measurements can focus on sites and hosts and can be defined on different time granularities. This capability will be used by the GridBench components in order to establish the properties of the high-level GridBench performance model. Output of the G-PM measurements will be provided to the user via the X-Window interface of G-PM, WP2.4.

5. INTERFACE DESCRIPTION

5.1. PORTALS

For a description of the interface with “Portals” refer to the “dependencies” section for Portals. (Section 4.2)

6. DETAILED DESIGN

Components that will be provided by Task 2.3 are logically separated into *GridBench* and *Utility Components.*

6.1. GRIDBENCH

GridBench will be comprised of a set of benchmark groups, where each set corresponds to a different approach of capturing aspects of grid performance. This is summarized in Table 1.

	Micro-benchmarks	Micro-kernels	Application Kernels
CE/SE	gb_ce_lin gb_se_io		
SITE	gb_site_lmnet gb_site_mpbench	gb_site_hpl gb_site_nas_is gb_site_nas_ep gb_site_nas_ft gb_site_nas_bt gb_site_nas_sp gb_site_nas_lu	[Task1.1] Kernels [Task1.2] Kernels [Task1.3] Kernels [Task1.4] Kernels
GRID	gb_grid_net	gb_grid_lin gb_grid_npb	gb_grid_cg

Table 1: Categorization of benchmarks

The GBBenchmark interface (outlined in Figure 2) will be implemented by all benchmarks, regardless of category.

Sections 0-6.1.7 make up a list of benchmark *sets.* A description is provided for each set. Specific designs for each of the benchmarks included in Table 1 will be produced as stated in the SRS[2].

1. Design of Micro-Benchmarks (gb_ce_lin, gb_se_io, gb_site_lmnet, gb_site_mpbench and gb_grid_net) by the end of **Month 8**.
2. Definition of representative micro-kernels by the end of **Month 10**.
3. Definition of a benchmark representative of a selected WP1 application (probably HEP) by the end of **Month 10**.

6.1.1. Micro-benchmarks at the CE/SE level:

Computing Elements and Storage Elements serve different purposes and measurements will be made based on their functionality.

CE's will be measured for Floating Point arithmetic performance on a realistic problem such as solving a set of linear equations. The metric to be measured is Flops (floating operations per second). Initially, CE performance will be measured based on a LINPACK kernel and a HEP kernel.

SE's will be measured for Disk I/O. The metric for SE's will be MB/s and it will be measured **(i)** as disk bandwidth that can be achieved by an application running locally, **(ii)** as bandwidth delivered to another local CE and **.(iii)** as bandwidth delivered to a CE at another site.

Components:

`gb_ce_lin`	A benchmark targeted at CE's that is based on the LINPACK[5] benchmark.
`gb_ce_hep`	A benchmark targeted at CE's that is based on the High Energy Physics kernels.
`gb_se_io`	A benchmark targeted at SE's for performing IO.

6.1.2. Micro-benchmarks at the Site level:

At the Site level (i.e. the cluster level) there are several subsystems that can be measured directly; Network, Intra-site MPI and CPU will initially be focused upon.

The most important network parameters are latency and bandwidth, and as such they will be measured. For latency, which is the time required for a small message to be delivered from one location to another, measurements need to be taken for the main protocols (such as TCP, UDP, ICMP and RPC). The unit of these measurements is of course the second. Bandwidth will be measured in B/s by employing larger or smaller messages over TCP.

The performance of Intra-site MPI will be measured for several commonplace MPI operations. While network measurements will be somewhat proportional to MPI measurements, MPI measurements need to be performed to take into account the more complex communication and overhead introduced by MPI. The intent is not to exhaustively measure MPI performance, but to determine the efficiency of MPI (by comparing with network benchmarks) or troubleshoot.

On the CrossGrid testbed each site will consist of several CE's, the processing power of which will be measured. The measurement will be collective, and the unit will be flops. The methodology is almost identical to the measurement of individual CE's but it is applied to all CE's concurrently.

Components:

`gb_site_lmnet`	Network benchmarks based on LMBench[6]
`gb_site_mpbench`	MPI benchmarks based on MPBench[7].

6.1.3. Micro-kernels at the Site level:

While the micro-benchmarks attempt to capture the performance characteristics of a site, they do so at such a low-level that it is almost impossible to extrapolate or predict performance. Micro-kernel benchmarks are included because they employ well-known codes and can provide results that can be compared with other high-performance computers. Candidate Micro-kernels that will be employed are:

LINPACK. A benchmark based on solution of equations based on the Basic Linear Algebra Subroutines[5]. More specifically, High-Performance LINPACK that is the scalable reincarnation. Selected because of the huge recognition, and its involvement in the ranking of the TOP500 [8] supercomputers. Provides measurements in flops.

Selected kernels from the NAS Parallel Benchmarks[9, 10]. These benchmarks were chosen because of their wide acceptance and real-world scientific kernels. Provides measurements in flops.

Components:

`gb_site_hpl` Linear equation solver based on the High Performance LINPACK[5] benchmark.

`gb_site_nas_is` Integer Sort based on the NAS Parallel Benchmarks[9, 10].

`gb_site_nas_ep` An Embarrassingly Parallel benchmark based on the NAS Parallel Benchmarks[9, 10]

`gb_site_nas_ft` A fast Fourier Transform benchmark based on the NAS Parallel Benchmarks[9, 10].

`gb_site_nas_bt` A Block Tridiagonal solver benchmark based on the NAS Parallel Benchmarks[9, 10].

`gb_site_nas_sp` A Pentadiagonal Solver benchmark based on the NAS Parallel Benchmarks[9, 10].

`gb_site_nas_lu` A LU solver benchmark based on the NAS Parallel Benchmarks[9, 10].

6.1.4. Application kernels at the Site level:

At the time of writing this, WP1 applications have not released extensive details of implementation or details of computational components that are to be employed. WP1 consists of 4 Tasks, each with intents to employ one or more computational kernels resulting in a rather large list of kernels. It is planned that the HEP kernel will be the pilot application for which benchmarks will be developed.

By reviewing the SRS's provided by WP1 this preliminary list of computational kernels has been derived:

1. [T1.1] Lattice-Boltzmann kernel
2. [T1.2] Meteorological kernel
3. [T1.2] Hydrological kernel
4. [T1.2] Hydraulic kernel
5. [T1.3] HEP kernel
6. [T1.4] Sea wave models (WAM4,SWAN)
7. [T1.4] Weather prediction kernel

Specific details and design for benchmarks based on some of these kernels will be provided when the WP1 tasks make their code available.

Measurements on the selected application kernels will be i) completion time in seconds , ii) flops and other higher-level application that are to be determined.

As a first phase (regarding application kernels) efforts will be focused on the HEP kernels because of their availability.

Components:

`gb_site_hep` A site benchmark based on the HEP kernels.

6.1.5. Micro-benchmarks at the Grid/Middleware level:

Though it may sound like a contradiction in terms micro-benchmarks at the Grid level, they will prove invaluable in interpreting kernel-based benchmarks at the Grid level (see next paragraph). Micro-benchmarks have the function of measuring one aspect of a system at a time. The Grid being the system at hand, networking and middleware is intuitively where the micro-benchmarking effort will concentrate. (The rest of the Grid components are not overlooked since most are captured in Site benchmarks). Networking at the Grid level has several functions (such as communicating results, inter-site MPI, file transfers etc.). Micro-benchmarks will measure latency, bandwidth. Latency (just as in the case of Site benchmarks) will be measured for the main protocols (such as TCP, UDP, ICMP and RPC) the second being the unit of measurement. Bandwidth will be measured in B/s by employing larger or smaller messages over TCP. For the Middleware part it is rather early to propose a comprehensive list of measurements because firstly the middleware performs many functions (in need of review) and secondly because it is difficult to say which ones are the really important ones that apply themselves to benchmarking. It is the purpose of the suite to stress test API's exported by the middleware to the applications (e.g. GridFTP). Initially Middleware micro-benchmarks will concentrate on Inter-site MPI and large file transfers.

Inter-site MPI benchmarks will very much resemble those of intra-site MPI. The distinction lies in that inter-site MPI communication will cross several boundaries of large differences in network performance. Since the placement of MPI processes (mapping) onto the grid is expected to drastically impact results, inter-site MPI benchmarks will need to account for this.

Large file transfers are expected to be common among many Grid applications. GridBench will provide benchmarks to test large file transfers through GridFTP.

Components:

`gb_grid_net`	A Grid-oriented Network benchmark.
`gb_grid_mpi`	A Grid-oriented MPI benchmark.
`gb_grid_ftp`	A benchmark for GridFTP.

6.1.6. Micro-kernels at the Grid level:

The use of micro-kernels for benchmarking Grid constellations will provide insight to the applicability of the grid to certain types of problems. They will also provide grounds for comparison to alternative architectures, given the wide acceptance/knowledge of the kernels to be employed. The Grid-enabled versions of these benchmarks will build directly on top of their site-level counterparts. A grid-version of the LINPACK benchmark as well as a subset of the NAS Parallel Benchmarks is included. The NPB subset will be based on the benchmarks selected for the site level. It is only included as a single benchmark (gb_grid_lin) since it will contain numerous combinations of benchmarks and data-flows.

Components:

`gb_grid_lin`	A Grid-oriented benchmark based on LINPACK (HPL)[5, 8].
`gb_grid_npb`	A Grid-oriented set of benchmarks that utilizes combinations of the NPB benchmarks[9, 10].

6.1.7. Application kernels at the Grid level:

Application kernel benchmarks are intended to provide practical tools for, among other things, application performance prediction, scalability and feasibility studies for running specific applications on specific Grid setups. As in the case of gb_grid_npb, gb_grid_cg will contain several kernels that are to be derived from CrossGrid applications. Again the HEP kernels will be considered first.

Components:

`gb_grid_cg` A Grid-oriented set of benchmarks that utilizes kernels from CrossGrid applications.

6.2. UTILITY COMPONENTS

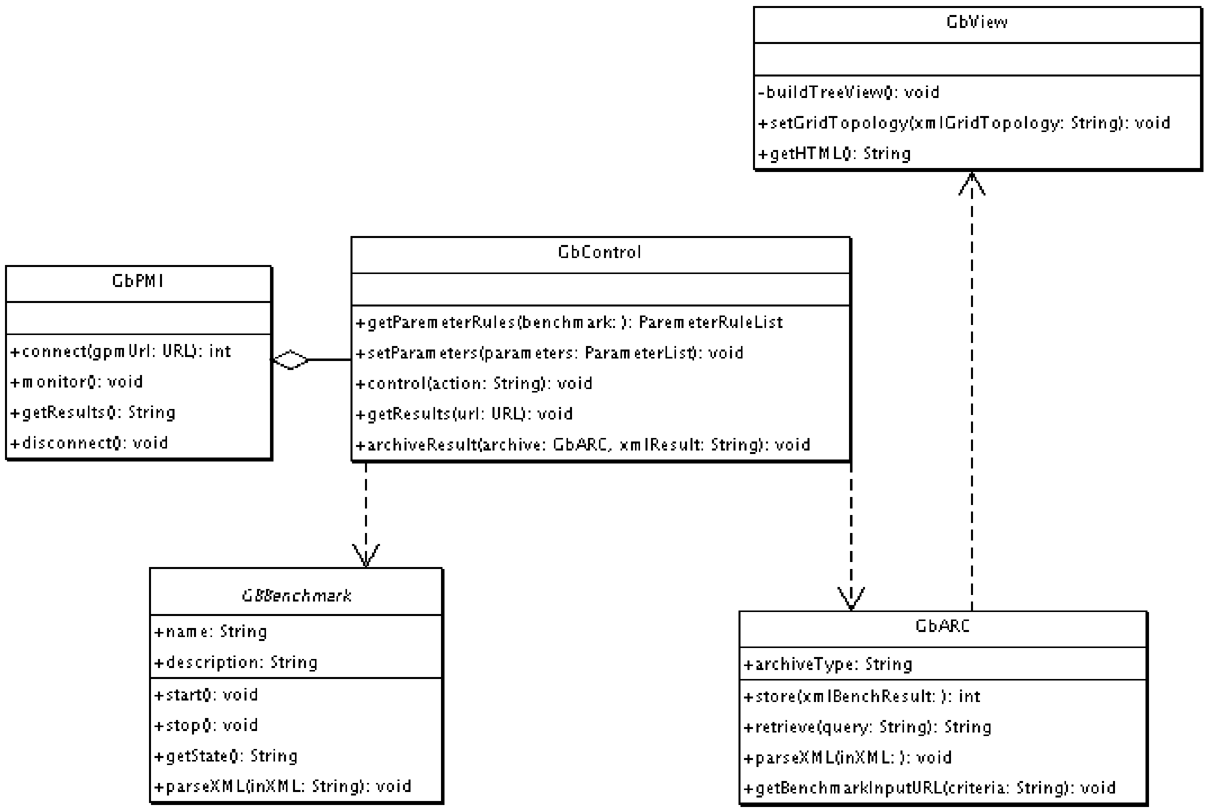
GbView
-buildTreeView(): void
+setGridTopology(xmlGridTopology: String): void
+getHTML(): String
GbPMI
+connect(gpmUrl: URL): int
+monitor(): void
+getResults(): String
+disconnect(): void
GbControl
+getParemeterRules(benchmark:): ParemeterRuleList
+setParameters(parameters: ParameterList): void
+control(action: String): void
+getResults(url: URL): void
+archiveResult(archive: GbARC, xmlResult: String): void
GBBenchmark
+name: String
+description: String
+start(): void
+stop(): void
+getState(): String
+parseXML(inXML: String): void
GbARC
+archiveType: String
+store(xmlBenchResult:): int
+retrieve(query: String): String
+parseXML(inXML:): void
+getBenchmarkInputURL(criteria: String): void

Figure 5: Utility components of GridBench

6.2.1. gbControl

General Description:

gbControl is the interface for running benchmarks both by the portal of T3.1 and by the user directly. gbControl will be usable from both a programmatic and command-line user interface. It can be thought of as a "wrapper" around the benchmark suite. (See Figure 5)

Functionality:

1. Provide parameter information for the generation of input forms (web or other) for the GridBench suite benchmarks.
2. Provides an interface for setting parameters and launching benchmarks.
3. Collects results (directly and/or through gbPMI).

4. Interfaces with gbARC.

6.2.2. gbPMI

General Description:

gbPMI is the GridBench Performance Monitoring Interface with G-PM. It will use the G-PM API to collect performance data during benchmark runtime. (See Figure 5)

Functionality:

1. Collect data through a G-PM API.
2. Make this data available to gbControl

6.2.3. gbARC

General Description:

gbARC is the Archival-Retrieval Component that interfaces to an underlying database with historical benchmarking data. The underlying database may be an LDAP server or (depending on the volume/nature of the results) an RDBMS. gbARC will be implemented as a service and will be based on HTTP. (See Figure 5)

Functionality:

1. Receive XML-encoded results of benchmarks and store them in the underlying DB.
2. Deliver XML-encoded results based on (potentially) complex queries, derived from the historical benchmarking data in the underlying DB.
3. Satisfy requests for XML-encoded META-DATA (regarding benchmark input files)

6.2.4. gbView

General Description:

gbView will serve visualization purposes for the benchmark results. (See section 4.2.2 for description)

Functionality:

1. Provide a browsable browse hypertextual interface for benchmark results (requested and historical)
2. Provide an interactive GUI for benchmark results (requested and historical)

7. REFERECES

[1]. Armstrong, B. and R. Eigenmann, *A Methodology for Scientific Benchmarking with Large-Scale Applications.*

[2]. Dikaiakos M., T.G., *WP2.3 SRS: GridBench, The CrossGrid Benchmark Suite.* June, 2002.

[3]. Funika W., R.F., Wismuller R., *WP2.4: Interactive and semiautomatic performance evaluation tools.* March, 2002.

[4]. *APART Specification Language (ASL).*

[5]. Jack Dongarra, P.L. and A. Petitet, *The LINPACK Benchmark: Past, Present, and Future.* December, 2001.

[6]. McVoy, L.W. and C. Staelin, *lmbench: Portable Tools for Performance Analysis*, in *USENIX Annual Technical Conference*. 1996. p. 279-294.

[7]. Mucci, P. and K. London, *The MPBench Report.* 1998.

[8]. Dongarra, J.J., H.W. Meuer, and E. Strohmaier, *TOP500 Supercomputer Sites, 11th Edition.* 1998(UT-CS-98-391).

[9]. David Bailey, T.H., William Saphir, Rob van der Wijngaart, Alex Woo, Maurice Yarrow. *The NAS Parallel Benchmarks 2.0.* in *The International Journal of Supercomputer Applications*. 1995.

[10]. Bailey, D.H., et al., *The NAS Parallel Benchmarks.* The International Journal of Supercomputer Applications, Fall, 1991. **5**(3): p. 63--73.