DELIVERABLE D2.3

PART III: PROTOTYPE DOCUMENTATION FOR GRIDBENCH

Task 2.2 Benchmarks and Metrics

Document Filename:	**CG2.3-D2.3-v1.0-UCY007-PrototypeDoc.doc**
Work package:	**WP2 Grid Application Programming Environment**
Partner(s):	**UCY**
Lead Partner:	**UCY**
Config ID:	**CG2.3-D2.3-v1.0-UCY007-PrototypeDoc**
Document classification:	**CONFIDENTIAL (to change go to File->Properties)**

Abstract: This document covers the Task2.3 Benchmarks and Metrics (GridBench) prototype. For the purpose of this prototype, one of the component benchmarks in the GridBench suite was selected for implementation.

Delivery Slip

	Name	Partner	Date	Signature
From				
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
1.0	16/01/2003	First Draft	Tsouloupas George, Dikaiakos Marios

CONTENTS

1.EXECUTIVE SUMMARY

This document covers the Task2.3 Benchmarks and Metrics (GridBench) prototype. For the purpose of this prototype, one of the component benchmarks in the GridBench suite (see design document) was selected for implementation. The selected benchmark is gb_site_hpl.

Gb_site_hpl is based on the well-known "High Performance Linpack" benchmark, which is used to rank the Top500 supercomputers. The HPL benchmark measures the performance of computers in solving a system of linear equations. The HPL has been modified by the implementation of a new XML based interfacing mechanism.

While the final version of this benchmark will provide much more detailed results, the prototype version will only provide a limited number of metrics. The metrics that will be provided by the prototype will be FLoating point OPerations pes Second (FLOPS) and completion time.

From the viewpoint of dependencies, the application assumes a working MPI installation on the target machines and availability of the BLAS library.

Installation is easy since it is RPM-based. Invocation of the benchmark is simple and straight forward with the provided default XML input file. The XML input file can be used to vary the problem size to fit the target machine and fine-tune the computation algorithm for accurate measurement. The output (i.e. the measurements) is given in an XML file.

2.INTRODUCTION

2.1.PURPOSE

The purpose of this document is to describe the prototype for Task 2.3 Benchmarks and Metrics. This document includes a general description of the benchmark that has been chosen to serve as a prototype. The document also includes a description of the input (parameter) file. Directions are given for building an RPM package and invoking the benchmark.

2.2.DEFINITIONS, ABBREVIATIONS, ACRONYMS

BLAS	Basic Linear Algebra Subprograms
CrossGrid	The EU CrossGrid Project IST-2001-32243
GridBench	The CrossGrid Benchmark Suite
FLOPS	FLoating point OPerations per Second
HPL	High Performance Linpack
MPI	Message Passing Interface
RPM	Redhat Package Management
XML	eXtesible Markup Language

3.REFERENCES

[1] CrossGrid Deliverable D2.3, Task 2.3 Design Document. June 2002. CG-2.3-D2.2-v1.1-UCY005-DesignDocument.doc

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

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

3.1.SOURCE CODE

The prototype will be referred to as "gb_site_hpl_proto" for the remainder of this document. Gb_site_hpl_proto is the prototype for the gb_site_hpl benchmark of the GridBench suite [1]. The gb_site_hpl is itself based on the High Performance Linpack benchmark [2,3]. Detailed source code documentation for the HPL benchmark can be found at http://www.netlib.org/benchmark/hpl/documentation.html.

3.2.CONTACT INFORMATION

Task Leader: Dr. Marios Dikaiakos (mdd@ucy.ac.cy)

George Tsouloupas (georget@ucy.ac.cy)

4.IMPLEMENTATION STRUCTURE

The Design Document describes several components that are aimed at collecting and archiving information from the benchmark executions. It is planned that in the final version of the software information collection will consist of:

1. Runtime data (I.e. information on the execution progress and metrics during the run of the benchmark);
2. Data (in the form of metrics, such as FLOPS) measured by the benchmarks on completion of the benchmark run.

For the purposes of this prototype only data returned by the benchmark on completion of its run will be collected. Many of the components included in the design of GridBench make little sense outside a grid environment, thus they are not included in this first prototype that is targeted at single local testbeds.

The Design Document introduced several benchmarks, of which one has been selected to serve as a prototype. The gb_site_hpl benchmark (which is based on the HPL benchmark, see the "references" section) has been selected to serve as a prototype. This decision was based on the fact that this benchmark is neither overly complex nor overly simplistic. The decision was also based on the fact that most of the source code of the application (with the exception of the interfaces) is available and quite thoroughly tested (HPL is quite a popular benchmark). Another reason for selection is that this benchmark can be used with no dependency on grid middleware.

The implementation of this prototype does not alter the design outlined in the Design Document since gb_site_hpl is a self-contained component. All work done for this prototype will directly contribute towards the finished system.

The following figure (figure 1) is taken from the Design Document; the implemented component is indicated by the red circle. The work done in terms of interfaces for gb_site_hpl is reusable for most of the other component benchmarks, especially "site-level" benchmarks (where the name is *_site_*).
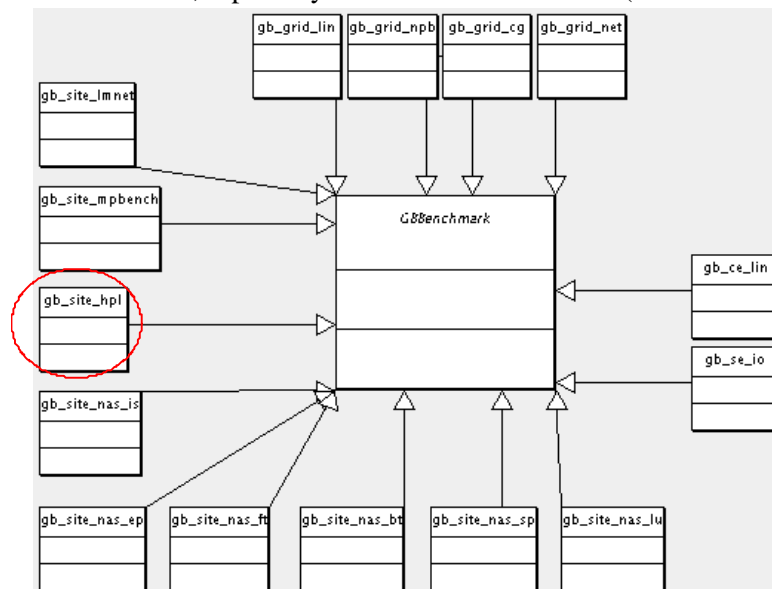

Figure 1

5.PROTOTYPE FUNCTIONALITY

5.1.ABOUT THE LINPACK BENCHMARK

The LINPACK benchmark solves a dense system of linear equations by Gaussian elimination. It stemmed out of an effort to assist users of the LINPACK package in estimating execution time of their applications [2]. By publishing the execution times of the solution to a fixed size 100x100 on several computers, users of the LINPACK library could extrapolate on their own application's performance. A formula of the following type was used to estimate the required time to process an n x n matrix:

$$time_n = \frac{time_{100} \cdot n^3}{100^3}$$

The solution of the system of equations is based on an LU decomposition with partial pivoting. At the time a 100x100 matrix was "large enough" but since then, CPU caches have become much bigger than that, thus a new scalable version has been developed: HPL (High Performance Linpack)[2]. Linpack is one of the most widely known benchmarks in HPC, HPL now ranks the TOP500 [3] super-computers. HPL can utilize either of two libraries, the Basic Linear Algebra Subprograms or the Vector Signal Image Processing Library, to perform the basic computations. The BLAS also formed the base of the LINPACK library package.

The application (benchmark) is fairly simple which, along with its portability, accounts for the most part of its acceptance; it consists of a component that solves the system of equations using an nxn matrix. This component relies on a set of BLAS routines that can be tailored and tuned to specific architectures thus making them efficient.

To get an idea of the operation count (64bit floating point operations) a tabulation of operation type and operation count is included as *Table 1*. While analysis of the underlying math and memory access patterns are essential for obtaining a good understating of the computation, the operation counts can provide a "feel" for the amount of computation. The operation counts along with completion times and theoretical CPU capabilities for many CPUs can be found in [2]

Table 1: Double precision operation counts
for the LINPACK 100 benchmark

Operation Type	Operation Count
Add	338250
Multiply	343200
Reciprocal	99
Divide	100
Negate	100
Absolute Value	5364
$\leq$	4950
$\neq 0$	5446
Total	697509

5.2.GB_SITE_HPL_PROTO FUNCTIONALITY

The gb_site_hpl_proto benchmark is a monolithic application, with a fairly simple user interface. It takes as input an XML file, which conforms to the schema given in the appendix. The output is again an XML file that contains two simple metrics: FLOPS and completion time. Using gb_site_hpl_proto is quite simple:

1. The user specifies the parameters of the benchmark (problem size, algorithm details, number of processors etc.) via the XML input file.
2. The benchmark as an ordinary MPI application.
3. The output is stored in an XML file.

The final gb_site_hpl will differ from gb_site_hpl_proto in the following:

1. The input parameters of the benchmark will be extended to accommodate monitoring, collection of intermediate results and running on several distributed resources
2. The output will be more detailed (more runtime data / intermediate results).

6.USER MANUAL

6.1.DEPENDENCIES

Having minimal dependencies for this prototype is important. There are two dependencies:

1. A dependency on BLAS (Basic Linear Algebra Subprograms) to which gb_site_hpl_proto must be linked. More precisely, the C interface (libcblas) is currently required.
2. A working MPI environment.

6.2.INSTALLATION

Gb_site_hpl_proto is installable by RPM:

```
rpm  –Uvh  gb_site_hpl-0.9.rpm
```

If the binary rpm is not available it can be built from source:

```
cd <path>/gb_site_hpl_proto
make rpm
```

The RPM is to be included in the next CG release and automatically installed in the User Interface machines. The code for the prototype is currently in the CrossGrid CVS: **cvs.fzk.de .**

6.3.RUNNING

To run gb_site_hpl_proto the user must first generate an input XML document (or use/edit the default one). A sample configuration file is default.xml. It contains valid input and should give reasonably accurate results (in terms of flops). (see section 6.4 for more information on input parameters).

The application is started (as a regular MPI application) by:

```
mpirun –np n gb_site_hpl -in in.xml –out out.xml
```

-np **n**	**N** is the number of processors (There are limitations on **n** imposed by the algorithm, see the interface description).
-in **infile**	**infile** is the input file (default is bg_site_hpl_in.xml)

`-out` ***outfile***	***outfile*** is the output file (default is `bg_site_hpl_out.xml`)

The output will be placed in the XML file specified above.

The benchmark can also be invoked by passing a single command-line parameter to it. In this case, the command-line parameter is the filename to a human-readable (ie. non-xml) initialisation file which is semantically similar to the xml input described. In this case the output will be directed to the standard output.

The benchmark can be executed by submitting RSL to a CE that supports MPI. For example, to run the benchmark on a CE "apelatis.grid.ucy.ac.cy" the following RSL can be submitted:

```
+
( &(resourceManagerContact="apelatis.grid.ucy.ac.cy:2119/jobmanager-pbs")
   (count=2)
   (label="HPL JOB")
   (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
                (LD_LIBRARY_PATH /opt/globus/lib/:/usr/local/globus/lib/)
           (GLOBUS_GRAM_JOB_CONTACT
apelatis.grid.ucy.ac.cy:2119/jobmanager-pbs))
   (arguments= "./xhpl.dat")
   (executable="./gb_site_hpl")
)
```

This assumes that the exacutable **`gb_site_hpl`** and the input file are already copied to the the directory of the user (on the CE).

6.4. INTERFACE DESCRIPTION

This section defines the input interface to gb_site_hpl_proto. The main XML tags are given and a short explanation. (The full XML schema as well as an xml example are given in the appendix). This information is adapted from http://www.netlib.org/benchmark/hpl/tuning.html, which includes information about tuning HPL through its legacy input file.

`<outputFileName> </outputFileName>`

The output filename

`<outputDevice> </outputDevice>`

This line specifies where the output should go. The line is formatted, it must begin with a positive integer, the rest is unsignificant. 3 choices are possible for the positive integer, 6 means that the output will go the standard output, 7 means that the output will go to the standard error. Any other integer means that the output should be redirected to a file

`<problemSizes> 3000,6000,10000  </problemSizes>`
This specifies the problem sizes one wants to run (up to 20).

```
<blockSizes> 80,100,120,140,160 </blockSizes>
```
This specifies the block sizes one wants to run (up to 20). The example means that one wants gb_site_hpl to use 5 differentblock sizes, namely 80, 100, 120, 140 and 160.

```
<processGrids>
      <pair>
             <P>1</P>
             <Q>6</Q>
      </pair>
      <pair>
             <P>2</P>
             <Q>8</Q>
      </pair>
      <pair>
             <P>4</P>
             <Q>13</Q>
      </pair>
</processGrids>
```
The processGrids tag contents indicate the number of process grids (up to 20) and specifically which process grids to use for the run. The above means that one wants to run gb_site_hpl_proto on 3 process grids (3 <pair> tags), namely 1-by-6, 2-by-8 and 4-by-13.

Note: In this example, it is required then to start gb_site_hpl_proto on at least 52 nodes (max of P-by-Q). The runs on the three grids will be consecutive. If one was starting gb_site_hpl_proto on more than 16 nodes, say 64, only 6 would be used for the first grid (1x6), then 16 (2x8) would be used for the second grid and 52 (4x13) would be used for the third grid.

```
<threshold>16.0</threshold>
```
This specifies the threshold to which the residuals should be compared with, for validation of numerically correct computation.

The remaining lines allow the user to specify algorithmic features. Gb_site_hpl_proto will run all possible combinations of those for each problem size, block size, process grid combination. This is handy when one looks for an "optimal" set of parameters.

```
                <pfacts>
                       <pfact>0</pfact>
                       <pfact>1</pfact>
                       <pfact>2</pfact>
                </pfacts>
                <nbmins> 1,2,4,8 </nbmins>
                <ndivs name="ndiv" type="int">
                </ndivs>
                <rfacts num="3">
                       <rfact>0</rfact>
```

```
                    <rfact>1</rfact>
                    <rfact>2</rfact>
            </rfacts>
            <bcasts num="2">
                    <bcasts>0</bcasts>
                    <bcasts>4</bcasts>
            </bcasts>
            <depths num="2">
                    <depths>0</depths>
                    <depths>1</depths>
            </depths>
            <swap num="3">
                    <swap>0</swap>
                    <swap>1</swap>
                    <swap>2</swap>
            </swap>
```

The panel factorization is matrix-matrix operation based and recursive, dividing the panel into **ndivs** sub panels at each step. This part of the panel factorization is denoted below by "recursive panel fact. (**rfact**)". The recursion stops when the current panel is made of less than or equal to **nbmin** columns. At that point, gb_site_hpl_proto uses a matrix-vector operation based factorization denoted below by **pfacts**.

```
<bcasts num="2">
      <bcasts>0</bcasts>
      <bcasts>4</bcasts>
</bcasts>
```

In the main loop of the algorithm, the current panel of column is broadcast in process rows using a virtual ring topology. HPL offers various choices and one most likely want to use the increasing ring modified encoded as 1. 3 and 4 are also good choices.

Valid inputs are 0-5: (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)

```
<depths num="2">
      <depths>0</depths>
      <depths>1</depths>
</depths>
```

These allow to specify the look-ahead depth used by HPL. A depth of 0 means that the next panel is factorised after the update by the current panel is completely finished. A depth of 1 means that the next panel is immediately factorized after being updated. The update by the current panel is then finished. A depth of k means that the k next panels are factorized immediately after being updated. The update by the current panel is then finished.

```
<swap num="3">
      <swap>0</swap>
      <swap>1</swap>
      <swap>2</swap>
</swap>
```

There are currently two swapping algorithms available, one based on "binary exchange" (0) and the other one based on a "spread-roll" procedure(1) (also called "long"). A value of two willidicate the a mixture of the two can be used (based on a threshold).

```
<lTransposed>0 </lTransposed>
```
Specifies whether the upper triangle of the panel of columns should be stored in no-transposed or transposed form.

```
<uTransposed>0 </uTransposed>
```
Specifies whether the panel of rows U should be stored in no-transposed or transposed form.

```
<equilibration>1</equilibration>
```
Enables(1) / disables(0) the equilibration phase. This option will not be used unless you selected 1 or 2 in *<swap></swap>*.

```
<memAlignment>8</memAlignment>
```
Specifies the alignment in memory for the memory space allocated by HPL. On modern machines, one probably wants to use 4, 8 or 16. This may result in a tiny amount of memory wasted.

The output of the prototype is a rather simple XML document, an example of which is given below:

```
<benchmark>
        <benchmarkName>gb_site_hpl</benchmarkName>
        <problemSize>1000</problemSize>
        <processes>2</processes>
        <CompletionTime>11.95</CompletionTime>
        <flops>55930000</flops>
</benchmark>
```

7.INTERNAL TESTS

The code on which gb_site_hpl_proto is based on is quite mature and tested. Modifications made to the software (i.e. interfaces) have been tested by comparing the input and output with the legacy input and output of HPL. Equivalent inputs gave identical output. The code was also thoroughly inspected.

8.ISSUES

There may potentially be a problem related to the dependencies of the benchmark, namely the BLAS. There is a libblas.a distributed with Datagrid (by CERN). The benchmark currently links with libcblas.a (the c interface to blas). Effort will be put to get it to link with the CERN libs, otherwise cblas will also have to be installed on the target machines.

There are currently no other known issues.

9.APPENDIX

The following is the XML schema to which input to gb_site_hpl_proto must conform. After the schema, an example xml file follows.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
     <xs:element name="P">
          <xs:simpleType>
               <xs:restriction base="xs:byte">
                    <xs:enumeration value="1"/>
                    <xs:enumeration value="2"/>
                    <xs:enumeration value="4"/>
               </xs:restriction>
          </xs:simpleType>
     </xs:element>
     <xs:element name="Q">
          <xs:simpleType>
               <xs:restriction base="xs:byte">
                    <xs:enumeration value="13"/>
                    <xs:enumeration value="6"/>
                    <xs:enumeration value="8"/>
               </xs:restriction>
          </xs:simpleType>
     </xs:element>
     <xs:complexType name="bcastsType" mixed="true">
          <xs:choice minOccurs="0" maxOccurs="unbounded">
               <xs:element name="bcasts" type="bcastsType"/>
          </xs:choice>
          <xs:attribute name="num" type="xs:byte"/>
     </xs:complexType>
     <xs:complexType name="benchmarkType">
          <xs:sequence>
               <xs:element ref="benchmarkName"/>
               <xs:element ref="description"/>
               <xs:element name="parameters" type="parametersType"/>
          </xs:sequence>
     </xs:complexType>
     <xs:element name="benchmarkName" type="xs:string"/>
     <xs:element name="blockSizes" type="xs:integer"/>
     <xs:complexType name="depthsType" mixed="true">
          <xs:choice minOccurs="0" maxOccurs="unbounded">
               <xs:element name="depths" type="depthsType"/>
          </xs:choice>
          <xs:attribute name="num" type="xs:byte"/>
     </xs:complexType>
     <xs:element name="description" type="xs:string"/>
     <xs:element name="equilibration" type="xs:boolean"/>
     <xs:element name="executableName" type="xs:string"/>
     <xs:element name="gridbenchmark">
```

```
            <xs:complexType>
                    <xs:sequence>
                            <xs:element name="benchmark" type="benchmarkType"/>
                    </xs:sequence>
            </xs:complexType>
    </xs:element>
    <xs:element name="lTransposed" type="xs:string"/>
    <xs:element name="memAlignment" type="xs:byte"/>
    <xs:element name="nbmins" type="xs:string"/>
    <xs:complexType name="ndivsType">
            <xs:simpleContent>
                    <xs:extension base="xs:string">
                            <xs:attribute name="name" type="xs:string"
use="required"/>
                            <xs:attribute name="type" type="xs:string"
use="required"/>
                    </xs:extension>
            </xs:simpleContent>
    </xs:complexType>
    <xs:element name="outputDevice" type="xs:integer"/>
    <xs:element name="outputFileName" type="xs:string"/>
    <xs:complexType name="pairType">
            <xs:sequence>
                    <xs:element ref="P"/>
                    <xs:element ref="Q"/>
            </xs:sequence>
    </xs:complexType>
    <xs:complexType name="parametersType">
            <xs:sequence>
                    <xs:element ref="executableName"/>
                    <xs:element ref="outputFileName"/>
                    <xs:element ref="outputDevice"/>
                    <xs:element ref="problemSizes"/>
                    <xs:element ref="blockSizes"/>
                    <xs:element name="processGrids" type="processGridsType"/>
                    <xs:element ref="threshold"/>
                    <xs:element name="pfacts" type="pfactsType"/>
                    <xs:element ref="nbmins"/>
                    <xs:element name="ndivs" type="ndivsType"/>
                    <xs:element name="rfacts" type="rfactsType"/>
                    <xs:element name="bcasts" type="bcastsType"/>
                    <xs:element name="depths" type="depthsType"/>
                    <xs:element name="swap" type="swapType"/>
                    <xs:element ref="lTransposed"/>
                    <xs:element ref="uTransposed"/>
                    <xs:element ref="equilibration"/>
                    <xs:element ref="memAlignment"/>
            </xs:sequence>
    </xs:complexType>
    <xs:element name="pfact">
            <xs:simpleType>
```

```
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:complexType name="pfactsType">
            <xs:sequence>
                <xs:element ref="pfact" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="problemSizes" type="xs:integer"/>
        <xs:complexType name="processGridsType">
            <xs:sequence>
                <xs:element name="pair" type="pairType"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="rfact">
            <xs:simpleType>
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:complexType name="rfactsType">
            <xs:sequence>
                <xs:element ref="rfact" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="num" type="xs:byte" use="required"/>
        </xs:complexType>
        <xs:complexType name="swapType" mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="swap" type="swapType"/>
            </xs:choice>
            <xs:attribute name="num" type="xs:byte"/>
        </xs:complexType>
        <xs:element name="threshold" type="xs:decimal"/>
        <xs:element name="uTransposed" type="xs:string"/>
</xs:schema>
```

An example of a conforming XML document follows:

```
<?xml version="1.0"?>
<gridbenchmark xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\xing-copy\schema\gridbench-ucy-wp3.1-
v1.xsd">
```

```
    <benchmark>
        <benchmarkName>gb_site_hpl</benchmarkName>
        <description>Adaptation of the High Performance LINPACK
Benchmark</description>
        <parameters>
            <executableName>xhpl</executableName>
            <outputFileName>HPL.out</outputFileName>
            <outputDevice>3,6,7</outputDevice>
            <problemSizes> 0,6,7  </problemSizes>
            <blockSizes> 80,100,120,140,160 </blockSizes>
            <processGrids>
                <pair>
                    <P>1</P>
                    <Q>6</Q>
                </pair>
                <pair>
                    <P>2</P>
                    <Q>8</Q>
                </pair>
                <pair>
                    <P>4</P>
                    <Q>13</Q>
                </pair>
            </processGrids>
            <threshold>16.0</threshold>
            <pfacts>
                <pfact>0</pfact>
                <pfact>1</pfact>
                <pfact>2</pfact>
            </pfacts>
            <nbmins> 1,2,4,8 </nbmins>
            <ndivs name="ndiv" type="int">
            </ndivs>
            <rfacts num="3">
                <rfact>0</rfact>
                <rfact>1</rfact>
                <rfact>2</rfact>
            </rfacts>
            <bcasts num="2">
                <bcasts>0</bcasts>
                <bcasts>4</bcasts>
            </bcasts>
            <depths num="2">
                <depths>0</depths>
                <depths>1</depths>
            </depths>
            <swap num="3">
                <swap>0</swap>
                <swap>1</swap>
                <swap>2</swap>
            </swap>
```

```
                <lTransposed>0 </lTransposed>
                <uTransposed>0 </uTransposed>
                <equilibration>1</equilibration>
                <memAlignment>8</memAlignment>
            </parameters>
        </benchmark>
</gridbenchmark>
```