



CONTRACT NUMBER 031513

**eDEISA**  
**EXTENDED DISTRIBUTED EUROPEAN  
INFRASTRUCTURE FOR  
SUPERCOMPUTING APPLICATIONS**

**European Community Sixth Framework Programme**  
**RESEARCH INFRASTRUCTURES**  
Integrated Infrastructure Initiative

Identification of an Initial Set of Application Codes and Low  
Level Benchmarks

Deliverable ID: eDEISA-D-eSA4-B1  
Due date : November 30 2006  
Actual delivery date: December 21, 2006  
Lead contractor for this deliverable: EPCC, UK

Project start date : June 1<sup>st</sup>, 2006  
Duration: 2 years

<b>Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)</b>		
<b>Dissemination Level</b>		
<b>PU</b>	Public	<b>X</b>
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	

## Table of Contents

Table of Contents .....	1
1. Introduction .....	2
1.1 Executive Summary.....	2
1.2 References and Applicable Documents .....	2
[3] IOzone Filesystem Benchmark – <a href="http://www.iozone.org/">http://www.iozone.org/</a> .....	2
1.3 Document Amendment Procedure .....	2
1.4 List of Acronyms and Abbreviations .....	2
2. Aims of the Benchmarking Activity .....	3
2.1 Choice of Applications Codes.....	3
2.2 Reporting Performance .....	4
2.3 Code Alterations.....	4
2.4 Licensing and Confidentiality.....	5
3. Benchmarking Strategy .....	6
3.1 Collection of Suggested Benchmarks.....	6
3.2 Initial Evaluation .....	7
3.3 Future work.....	7
4. Low-Level Benchmarks .....	9
4.1 HPC Challenge.....	9
4.2 IOzone and b_eff_io .....	9
4.3 Additional low-level benchmarks .....	10
5. Conclusions .....	10
6. Appendix A: List of candidate applications .....	11
7. Appendix B: Initial Application Questionnaire .....	12

## 1. Introduction

### 1.1 *Executive Summary*

The objective of the eDEISA benchmarking project (task 3 in the eSA4 Applications Enabling Service Activity) is to design and support a benchmark suite that can be used easily to evaluate the performance of high-end supercomputers. It will include a number of real applications codes and associated datasets which are representative of the research undertaken by Europe's leading computational scientists. It will also include a set of low-level synthetic benchmarks to evaluate the performance of key components of the machine architecture.

This document is a report on the initial activities of the benchmarking group undertaken during the first six months of the two-year eDEISA project.

Section 2 outlines the scope of the project and discusses a range of issues which are expected to be important in the construction of a suitable benchmark suite.

Section 3 describes the overall benchmarking strategy and presents the applications codes that comprise the proposed initial superset for the suite. It also describes the procedure by which this superset will be reduced to form the initial benchmark release (version 1) due at PM12, and the final release (version 2) due at PM24.

Section 4 covers the selection of low-level benchmarks.

Appendix A contains a full list of the superset of benchmark applications.

Appendix B contains the standard questionnaire template used for initial evaluation of the proposed benchmarks.

### 1.2 *References and Applicable Documents*

- [1] eDEISA Annex I – “Description of Work”, 27 March 2006.
- [2] HPC Challenge Benchmark – <http://icl.cs.utk.edu/hpcc/>
- [3] IOzone Filesystem Benchmark – <http://www.iozone.org/>
- [4] Effective I/O Bandwidth (b\_eff\_io) – [http://www.hlrs.de/organization/par/services/models/mpi/b\\_eff\\_io/](http://www.hlrs.de/organization/par/services/models/mpi/b_eff_io/)

### 1.3 *Document Amendment Procedure*

Reviewed internally by EPCC and eSA4-T3 team, and by external eDEISA reviewer.

### 1.4 *List of Acronyms and Abbreviations*

<b>CFD</b>	Computational Fluid Dynamics
<b>DECI</b>	DEISA Extreme Computing Initiative
<b>DEISA</b>	Distributed European Infrastructure for Supercomputing Applications
<b>QCD</b>	Quantum ChromoDynamics

## 2. Aims of the Benchmarking Activity

The aim of the benchmarking activity is to provide a single, coherent collection of software that can easily be used to evaluate the performance of high-end supercomputers. An ideal benchmarking suite would have all of the following characteristics:

- cover all major scientific applications areas
- include all major programming languages
- straightforward and quick to run
- stress all the key components of the machine architecture
- easy to port to new architectures
- self-verifying (ie automatically check results)
- report easy to interpret performance information
- highly scalable (in both numbers of processors and problem size)
- cover aspects of vector and scalar operations
- measure low-level and full application performance
- easily available to a wide audience at no cost

In practice, however, it is not possible to satisfy all these requirements and a number of compromises have to be taken. The issues that have to be considered in creating a practical (as opposed to ideal) benchmarking suite are outlined in the following subsections. It is essential to explain these points in some detail as they are all likely to be important matters that are encountered during the lifetime of the project.

### 2.1 Choice of Applications Codes

In order to be representative and accepted by the HPC community, the suite must cover a wide range of scientific applications. However, in order to be able to port and run the suite in a reasonable amount of time it must definitely contain less than ten codes: when comparing machine performance it is essential to have complete sets of measurements in all cases, so it is important that it is straightforward to run the entire suite. It is therefore not possible to cover all applications of HPC in Europe, so some selection criteria must be used.

A more realistic aim is to design a suite that provides sufficient information to enable the performance of a wide range of applications codes to be easily predicted, even if a particular application is not actually executed. This can be done in two ways:

- ensuring that the suite contains codes that cover the major computational techniques used in HPC in Europe
- running a sufficient range of low-level benchmarks to allow application performance to be predicted using parameterised performance models

Few scientists have sufficient time to develop accurate performance models of their codes, so in practice the first aim is most important. An example might be a fluid dynamics code and a quantum chemistry code that both rely heavily on parallel fast fourier transforms. Despite the totally different scientific research done by these codes the performance characteristics may be very similar due to their common use of FFTs, and therefore only one code may need to be included in the benchmark.

There is another situation where a single application can be used to inform two different scientific communities, which is when they use exactly the same applications code. This can be the case with, for example, molecular dynamics

packages where the same code can do a wide variety of calculations depending on the dataset. One calculation might be relevant to materials science, the other to the life sciences. In this case it is important to include multiple datasets in the benchmark as performance characteristics can vary dramatically depending on the details of the calculation being performed. However, it is much easier to port and run a single application with two input datasets than to have to deal with two independent applications codes.

In order to assess the scientific coverage required by the suite, we classify all proposed codes into the same categories of applications areas already used by the DEISA Extreme Computing Initiative (DECI):

- material science
- life sciences and informatics
- earth science and climate research
- astrophysics
- CFD and combustion
- plasma physics
- QCD

It will be an important test of the benchmark suite that it can be useful in all of the above areas.

## **2.2 Reporting Performance**

A scientist running a particular application with a particular dataset is only concerned with the end-to-end runtime of the entire program. However, as we cannot include all applications in the suite it is important that useful performance metrics are reported from the codes. This is straightforward for low-level benchmarks which have been specifically designed to measure quantities such as interconnect bandwidth or memory latency. However, it may be necessary to introduce additional performance measurements into applications codes to make them useful in the benchmark. A basic example would be to report IO and calculation timings independently. Taking the example mentioned previously of the CFD and chemistry codes that both rely on parallel FFTs, it would be important to ensure that the time taken by the FFT is reported separately in order that only one of them needs be included in the benchmark.

In a real research environment, scientists do not have exclusive access to a supercomputer. Their scientific productivity is therefore dependent on how quickly their jobs run from submission to completion and not just on the sustained GFlops of their code. To evaluate the overall performance of a supercomputer and its batch system under a mixed workload it is necessary to design a throughput benchmark that involves running a basket of different applications codes simultaneously. However, this makes the benchmarking procedure longer and more complicated so such throughput tests have to be very carefully designed.

## **2.3 Code Alterations**

There are a number of situations in which it may be necessary or desirable to alter various aspects of an application's source:

- code changes to enable compilation on different architectures
- code changes for optimal performance on different architectures
- inclusion of additional performance measurements

- inclusion of additional scientific measurements to aid verification
- changes to the build procedure (eg the Makefile) for consistency across the benchmark suite

All of these mean that the code that is being run is different from the code originally supplied by the scientific authors. This may cause problems and extra work in the future if the application is altered or enhanced, at which point a decision has to be taken as to whether the benchmark should be updated to reflect the current state-of-the-art or whether it should be frozen for backwards compatibility. When code is changed for performance reasons it is important to be able to judge if the alterations are acceptable. For example, completely rewriting a code for a new architecture is not acceptable as it does not correspond to something a computational scientist would want to do in practice. However, replacing source code by an optimised library would be acceptable providing it was done in a portable manner.

It would be tempting to consider producing a single consistent build procedure for all of the applications which, although quite a lot of work, would make the suite extremely easy to compile and run. However, such an approach would be almost impossible to maintain, so we will aim to extend the existing Makefiles where required (eg for new architectures) rather than rewrite them. To give a consistent look-and-feel, the best approach is probably to have a single higher-level configuration script that allows building the entire suite to be easily controlled from a single point.

In general we will aim to work with the code authors and feed back any changes or improvements so that they can be included in the release version if appropriate. However, we will need to draw up rules for the acceptability of source code changes for the situation where computer vendors choose to optimise applications for their own platforms.

## **2.4 Licensing and Confidentiality**

An enormous amount of intellectual effort and manpower goes in to designing and writing a scientific application code. As a result, code authors may naturally be unwilling to give their source for free to potential academic or commercial competitors. For example, some of the major applications packages have associated licensing agreements that we will need to take into account when distributing the benchmark suite.

The situation is clearer when the benchmark is used as part of a procurement exercise. In such a case, vendors are familiar with the procedure of signing temporary licence agreements and deleting the codes once they have been used. If the benchmark is to be made more generally available then this may restrict the range of applications codes that we can choose from. Even for non-licensed codes there may be issues with distributing source code that contains a scientist's current research work. In such a case we could consider constructing a version that has restricted functionality and is therefore less academically sensitive, but it is clear that for a public benchmark we need to work much more closely with the code authors than for a benchmark suite that is only used in commercial procurements.

We will need to strike a balance between the range of codes that we include in the benchmark and the range of people that can run it (and hence the range of platforms from which we obtain results).

### 3. Benchmarking Strategy

The activities of the benchmarking group are planned to follow the structure:

1. Collection of suggested benchmarks from DEISA partners
2. Evaluation of benchmark superset based on high-level criteria
3. Selection of initial benchmark suite (version 1) by DEISA Policy Committee
4. Initial investigation of benchmark v1 portability, performance and datasets
5. Initial benchmark release (version 1)
6. Detailed investigation of benchmark v1 portability, performance and datasets
7. Benchmark v1 updated based on results of detailed investigation
8. Selection of final benchmark suite (version 2) by DEISA Policy Committee
9. Final investigation of benchmark v2 portability, performance and datasets
10. Final benchmark release (version 2)

To meet the deliverable timescale set out in the Technical Annex, stage 1 must be complete by PM6; stage 5 by PM12; stage 7 by PM18 and stage 10 by PM24.

#### 3.1 Collection of Suggested Benchmarks

All DEISA partners were contacted and asked for input for the benchmark suite. They were asked to consider the following points when selecting their proposals:

- low-level and user benchmarks currently used at the partner site
- usage profile of each site by scientific area
- current DECI codes being supported
- codes associated with ongoing DEISA Research Activities

At a meeting of the group held in Edinburgh, presentations were given by all active benchmarking partners and the various issues discussed. This resulted in a total of 53 suggestions (42 different codes if duplicates are removed) listed in full in Table 2, Appendix A; this is considered to be a superset of the benchmarking suite, ie we expect that the final set of benchmarking applications will be taken from this list. The distribution by applications area is shown in Figure 1. This includes duplicates (eg if the same code is suggested by two sites then it is counted twice) as this gives a true reflection of the distribution of applications areas across the DEISA partners.

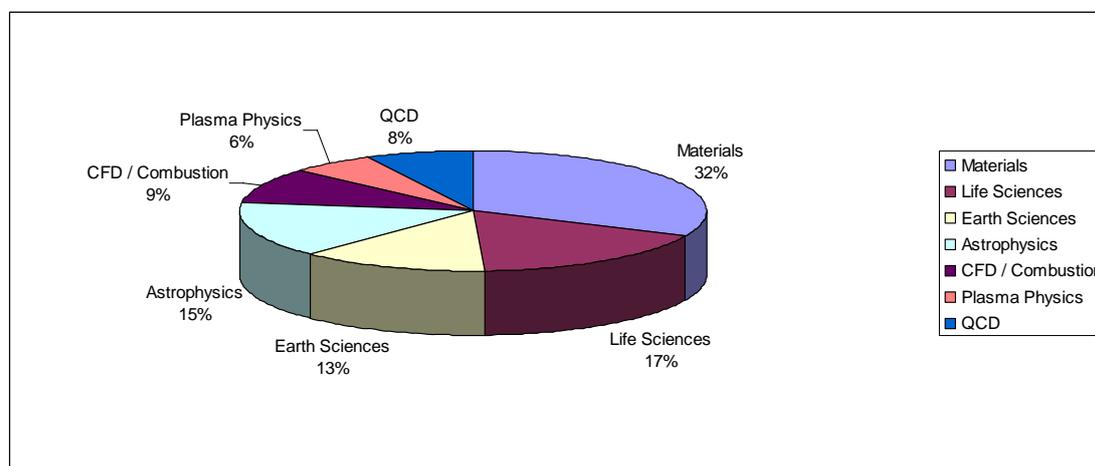


Figure 1: Distribution of benchmarking superset by applications area

Six codes were suggested by more than one partner, and these are listed in Table 1.

Code	Applications Area	Multiplicity
CPMD	Materials Science	4
GADGET	Astrophysics	3
NAMD	Life Sciences	3
VASP	Materials Science	3
BQCD	QCD	2
CASTEP	Materials Science	2

**Table 1: Statistics for codes suggested by multiple DEISA partners**

### 3.2 Initial Evaluation

Given the large number of codes in the superset it is important to produce a brief summary of each application which will enable them to easily be compared on an equal footing. To this end, a standard questionnaire was designed and distributed to all partner sites, a copy to be completed for each proposed application in the superset. The questions were designed to cover the following areas of interest:

- General description
- Scientific relevance
- Support available from authors
- Implementation
- Performance and scalability
- Portability

The full questionnaire is included in Appendix B; sample answers were included in the template to give guidance on the types of responses expected. So far we have completed 32 of the 53 questionnaires.

Once the entire set has been collected the eSA4-T3 team will conduct a review of all the candidate codes and assess them according to their suitability. This review will result in a ranking of codes within each scientific category which will be presented to the DEISA Policy Committee as the recommendations of the benchmarking team. This ranking will be accompanied by a short report that explains the reasons for the chosen ordering. The Policy Committee will then have access to a complete set of technical information about the candidates so that they can take a decision on the initial benchmark suite of around 15 codes. It will be up to the Policy Committee whether they simply wish to select codes from the list based on the distribution of applications areas given in Figure 1, or whether other strategic factors should also be taken into account.

### 3.3 Future work

We would expect that the initial suite will have about twice the number of codes expected in the final suite, ie will include no more than around fifteen codes. These will be distributed among benchmarking staff for a period of intensive porting and performance studies on the DEISA platforms. This will result in a clean set of portable codes and validated data sets that will form version 1 of the benchmark.

The next stage will involve work on designing and verifying much larger datasets than are currently available. This is because it is important that the benchmark is suitable for future machines that are substantially more powerful than even the largest current DEISA platforms. The existing datasets and problem sizes may not be

applicable to machines which are approaching a Petaflop of peak performance. We will produce a report on each code in version 1 and compare the results to identify the most suitable codes for the final benchmark. It is at this stage that we would try to identify codes with similar performance characteristics, which might allow one of them to be dropped. Note that such a comparison is only meaningful after these detailed studies have been conducted on a wide range of HPC platforms. Based on these reports, the final benchmark suite (version 2) of around six to eight codes will be selected and released.

At all stages in the process we will be open to evaluating new applications codes that may arise over time, with a view to possible inclusion in the benchmark suite. DEISA will work closely with the HPC organisations of member states that are preparing a scientific case for a future European HPC service. These activities will start as soon as the ongoing discussions have resulted in a definite roadmap for the structure of future European HPC centres. We will also establish relationships as appropriate with other benchmarking groups. For example, we will exploit the fact that ECMWF is already active in the RAPS benchmarking activity for climate research. The possibility of including industrial codes in the benchmark will be investigated through the strong industrial links of a number of the DEISA partners such as HLRS (automotive industry) and CINECA (industrial computational fluid dynamics).

## 4. Low-Level Benchmarks

Low-level performance benchmarks have been a matter of extensive research in HPC for many years. There are a number of widely used and well accepted benchmarks already in existence and we do not propose to spend substantial effort in developing new software in this area. We will use the standard HPC Challenge (HPCC) benchmarks to measure various key aspects of serial and parallel performance, and use either the IOzone or `b_eff_io` benchmarks to measure file access performance.

The HPCC and IO benchmarks cover the basic general features of a parallel machine's architecture. However, there are more detailed hardware characteristics that may also be of interest. As a result, we will leave the low-level benchmark suite open to additional tests, examples of which are outlined in section 4.3. Whether or not a test is included will be based on the results obtained from the investigations of applications performance. It is these applications results that will determine which detailed architectural characteristics are most relevant in practice to sustained performance.

### 4.1 HPC Challenge

The HPCC benchmark is a collection of seven standard low-level benchmarks which assess both serial and parallel performance.

- |                        |  |
|------------------------|--|
| 1. <b>HPL</b>          | The standard Linpack benchmark                       |
| 2. <b>DGEMM</b>        | Matrix-matrix multiplication (the core of HPL)       |
| 3. <b>STREAM</b>       | Bandwidth between CPU and main memory                |
| 4. <b>PTRANS</b>       | A saturation benchmark for the communications system |
| 5. <b>RandomAccess</b> | Scattered memory updates (serial and parallel)       |
| 6. <b>FFT</b>          | Serial Fast Fourier Transform                        |
| 7. <b>b_eff</b>        | Latency and bandwidth of various comms patterns      |

We propose to use this benchmark without modification in order to allow for easy comparison with other systems.

### 4.2 IOzone and `b_eff_io`

The IOzone benchmark measures a variety of file access operations including streaming and scattered reads and writes. Although it is fundamentally a serial benchmark, it can be used to test the parallel IO capabilities of a system by running multiple instances simultaneously on many nodes.

The `b_eff_io` benchmark is a suite designed to test parallel IO done using the `MPI_IO` interface. Although it is therefore much higher level than IOzone, its performance depends on both the capabilities of the underlying IO system and the quality of the `MPI_IO` implementation. This is also true of standard inter-processor communications benchmarks which depend on both the performance of the interconnect and of the MPI implementation. For communication, MPI is so pervasive in HPC that this is not an issue. However, `MPI_IO` is not so widely used in current HPC applications which usually implement their own methodologies for parallel IO. As a result it is not obvious how relevant the `b_eff_io` measurements are to real applications performance.

Running these benchmarks requires a substantial number of configuration choices due to the varied different characteristics of parallel IO systems. We will investigate both IOzone and b\_eff\_io on the DEISA platforms and choose a single candidate based on the results.

### **4.3 Additional low-level benchmarks**

As well as the aspects of performance covered by the HPCC and IO benchmarks, other architectural characteristics could also be important, for example:

- extended memory bandwidth (eg bandwidth from different levels of cache, strided access patterns, random access,... );
- system noise caused by interference between the application and the OS;
- ability to overlap communication with computation;
- performance of f95 intrinsic constructs;
- performance of particular MPI collectives routines (eg MPI\_Alltoallv).

For many of these, standard benchmarks already exist (eg PSNAP for system noise) but it might be necessary for us to develop small benchmark codes in particular cases. We will introduce additional low-level benchmarks as required, but will always take into account the necessity of keeping the size of the low-level suite at a manageable level.

## **5. Conclusions**

The benchmarking activity is proceeding well. The applications benchmarks will be taken from the superset list of 53 proposals (42 distinct codes) as detailed in Table 1 in Appendix A. The selection of the initial benchmark suite (expected to contain around 15 codes) will take into account criteria such as applications area, scientific importance and scalability as reported in the Application Questionnaires (see Appendix B). Selection of the final benchmark (expected to contain around 8 codes) will be based on more detailed and thorough investigations undertaken on the whole range of DEISA platforms. The low-level benchmarks will comprise the HPC Challenge suite, one IO benchmark (either IOzone or b\_eff\_io) and selected additional detailed benchmarks as required.

## 6. Appendix A: List of candidate applications

	Materials Science	Life Sciences + Informatics	Earth Sciences + Climate Research	Astro-physics	CFD + Combustion	Plasma Physics	QCD
<b>BSC</b>	CPMD	NAMD	WRF	GADGET			
<b>CINECA</b>	Quantum-Espresso	NAMD		GADGET			
<b>CSC</b>	VASP DALTON GROMACS	HMMER	HIRLAM	POLAR	POROUS		SU3_AHIGGS
<b>ECMWF</b>			IFS				
<b>EPCC</b>	VASP CASTEP HELIUM CASINO	DL_POLY NAMD	UM		PDNS3D		
<b>FZJ</b>	CPMD	IQCS SMMP		NBODY6++		PEPC	BQCD QCDOIB
<b>HLRS</b>					Fenfloss		
<b>IDRIS</b>			NEMO	RAMSES	AVBP		
<b>LRZ</b>	NWCHEM			CACTUS	BEST		BQCD
<b>RZG</b>	CPMD WIEN2K CASTEP ESPResSo		ECHAM5	GADGET SUCCEsS		ORB5 GENE	
<b>SARA</b>	VASP CPMD	GAMESS-UK ADF	TM 5				

**Table 2: Complete list of suggested codes by site and applications area**

## 7. Appendix B: Initial Application Questionnaire

Name of reviewer: eg David Henty

-----

Institution: eg EPCC

-----

Code Name:

- o eg a.out

Applications Area:

- o eg materials

Brief Description:

- o eg this is a classical MD code with empirical forces, designed in particular for simulations of organic molecules

Scientific Relevance:

- o give some measure of the national/international importance of the code

Support:

- o who are the authors of the code?
- o will we have support from the code authors?
- o is the code supported by a JRA or DECI project?

Implementation:

- o Language (C, C++, Fortran)
- o Estimate of lines of code (eg 50,000)
- o Parallelisation(s): MPI, MPI+OpenMP, other platform specific parallel model (such as SHMEM on Cray), ...

Performance and Scalability:

- o give data on the relative scaling efficiency as you increase the processor count by a factor of 2, eg if you have timings on 64, 128, 256 and 512 CPUs (T64, T128, T256 and T512) then you would compute three efficiencies (E128, E256 and E512):

$$E128 = T64/(2*T128)$$

$$E256 = T128/(2*T256)$$

$$E512 = T256/(2*T512)$$

- give information how the time is measured (total time, time per iteration etc) and what was the problem size / data set used; give information on different problem sizes / data sets / architectures if possible
- o list the the most important kernel(s) in terms of performance
  - eg FFT, eigenvalue calculation, IO, sparse linear solver, dense matrix operations, ...
- o what part(s) of the architecture will be stressed by the code
  - eg is it IO, interconnect (point-to-point, global reductions, all-to-all comms, ...), memory bandwidth, memory latency, ...

Portability:

- o what platform(s) does the code run on currently?
  - any know portability problems
- o what external libraries does it call (other than MPI)
  - eg FFTW, NetCDF, ScaLAPACK, charm++, ...