

Benchmarking Grid Applications

Farrukh Nadeem , Radu Prodan, Thomas Fahringer
Institute of Computer Science University of Innsbruck
`{farrukh, radu, tf}@dps.uibk.ac.at`

Alexandru Iosup
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology, The Netherlands.
`A.Iosup@tudelft.nl`



CoreGRID Technical Report
Number TR-0104
August 31, 2007

Institute on Grid Information, Resource and Workflow
Monitoring Services
Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: <http://www.coregrid.net>

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Benchmarking Grid Applications

Farrukh Nadeem , Radu Prodan, Thomas Fahringer
Institute of Computer Science University of Innsbruck
{farrukh, radu, tf}@dps.uibk.ac.at

Alexandru Iosup
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology, The Netherlands.

A.Iosup@tudelft.nl

CoreGRID TR-0104

August 31, 2007

Abstract

Application benchmarks can play a key role in analyzing and predicting the performance and scalability of Grid applications, serve as an evaluation of the fitness of a collection of Grid resources for running a specific application or class of applications [27], and help in implementing performance-aware resource allocation policies of real time job schedulers. However, application benchmarks have been largely ignored due to diversified types of applications, multi-constrained executions, dynamic Grid behavior and heavy computational costs. To remedy these, we present the GrapBench (**Grid Application Benchmarks**) system. GrapBench computes the Grap Benchmarks for Grid applications which are flexible regarding variations in problem-size of the application and machine-size of the Grid-site. GrapBench dynamically controls the number of benchmarking experiments for individual applications, and manages the execution of these experiments on different Grid-sites in an easy and flexible way. We also present results from the prototype implementation of our proposed system to show the effectiveness of our approach.

1 Introduction

Grid infrastructure provides an opportunity to the scientific and business communities to exploit the powers of heterogeneous resources in multiple administrative domains under a single umbrella [13]. Proper characterization of Grid resources is of key importance in effective mapping and scheduling of the jobs to these resources, (to utilize maximum power of these resources). Benchmarking has been used for many years to characterize a large variety of systems ranging from CPU architectures to the file-systems, databases, parallel systems, internet infrastructures, middlewares etc. [11]. There have always been issues in optimized mapping of jobs to the Grid resources on the basis of available benchmarks [26]. Existing Grid benchmarks (or their combinations) do not suffice to measure/predict application performance and scalability, and give a quantitative comparison of different Grid-sites for individual applications while taking into effect variations in the problem-size. In addition, there are no integration mechanisms and common units available for existing benchmarks to make meaningful inferences about the performance and scalability of individual Grid applications on different Grid-sites.

Application benchmarking on the Grid can provide a basis for users and Grid middleware services (like meta scheduler, resource broker) for optimized mapping of jobs to the Grid resources by serving as evaluation of fitness to compare different computing resources in the Grid. The performance results obtained from real application benchmarking are much more useful for running applications on a highly distributed Grid infrastructure than the regular

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

resource information provided by the standard Grid information services [26]. Application benchmarks are also helpful in predicting the performance and scalability of Grid applications, studying the effects of variations in application performance for different problem-sizes, and gaining insights into the properties of computing nodes architectures. However, the complexity, heterogeneity and the dynamic nature of Grids raise serious questions about the overall realization and applicability of application benchmarking. Moreover, diversified types of applications, multi-constrained executions, and heavy computational costs make the problem even harder. Above all, mechanizing the whole process of controlling and managing benchmarking experiments and making benchmarks available to users and Grid services in an easy and flexible fashion makes the problem more challenging.

To overcome this situation, we present GrapBench, a four layered Grid applications benchmarking system. GrapBench produces benchmarks for Grid applications taking into effect the variations in problem-size of the application and machine-size of the Grid-site, thus allowing problem-size and machine-size flexible comparison of Grid-sites for individual applications. GrapBench provides the necessary support for conducting controlled and reproducible experiments, for computing performance benchmarks accurately and for interpreting benchmarking results in the context of applications' performance and scalability predictions and application specific comparison of different Grid-sites through a graphical user interface. GrapBench takes the specifications of executables, set of problem-sizes, pre-execution requirements and set of available Grid-sites as an input in XML format. These XML specifications, along with the available resources are parsed to generate jobs to be submitted to different Grid-sites. At first, GrapBench completes pre-experiment requirements, if any, and then runs the experiments according to the experimental strategy. Benchmarks are computed from experimental results and archived in benchmarks repository for later use. Performance and scalability prediction and analysis from the benchmarks are available through a GUI and GT4 service interfaces. We do not require complex integration/analysis of measurements, or new metrics for interpretation of benchmarking results.

Among our considerations for the design of Grid application benchmarks were conciseness, portability, easy computation and adaptability for different Grid users/services. We have implemented a prototype of the proposed system under ASKALON [12], on the top of GT4 [2], which is a reference implementation of WSRF [30].

The rest of the paper follows as such: Section 2 describes performance of a Grid application and the different factors affecting it. The design of *benchmarks generation process* is described in the Section 3. In Section 4 we describe the number of benchmarks experiments and autoGrap a semi-automatic application benchmarking tool to make benchmarking experiments. Computation of benchmarks from experimental results is explained in Section 5. In Section 6 we describe utilization of benchmarks. We present results from experiments conducted on Austrian Grid and related analysis in Section 7. Related work is presented in Section 8, and finally we conclude in Section 9.

2 Performance of a Grid Application

Performance of an application on a Grid-site is dependent upon the performance of a couple of inter-related Grid resources at different levels of Grid infrastructure: performance of: the Grid-site, the computing node, CPU, memory hierarchy, I/O, storage node, network (LAN/WAN), network topology etc. as shown in Figure 1, adapted from [11]. Our conjecture is that, the traditional benchmarks (combination of benchmarks) and their context of use cannot be directly used for application performance prediction for multiple reasons: different performance representation of individual resources, high cost (with respect to time and money), trust worthiness of benchmarking suits and corresponding measurements, metrics interpretation, and above all the complex integration of results from different resources, to make some conclusions useful for human users and Grid middleware services. Moreover, existing sets of benchmarks do not allow *cross-platform interoperability* [9] of benchmarks results at different structural levels of the Grid, for different Grid applications. More specifically there is a need for benchmarks, which

- (i) represent the performance of Grid application on different Grid sites
- (ii) incorporate the individual effects of different Grid resources specific to different applications (like memory, caching etc.)
- (iii) can be used for performance and scalability predictions of the application
- (iv) are portable to different platforms
- (v) are flexible regarding variations in problem-size and machine-size
- (vi) support fast and simplified computation and management
- (vii) are comprehensively understandable and usable by different users and services

On the other hand, it is also necessary to address the high cost of Grid benchmarking administration, and benchmarking computation and analysis. This requires a comprehensive system for benchmarking computation, management, with a visualization and analysis component.

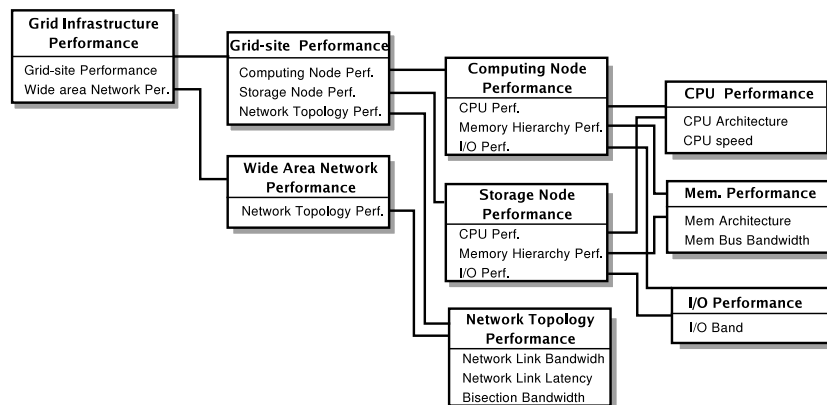


Figure 1: Different factors affecting application performance in the Grid.

3 GrapBench System Design

GrapBench benchmarks Grid applications on different Grid-sites. It consists of a framework containing set of tools to perform and facilitate the benchmarking process (the benchmarking experiments, computation and storage of results) in a flexible way, and later publishing the results and analysis.

The process of benchmarks generation is shown in the Figure 2. Layer 1 specifies application details for benchmarking experiments in XML based Grid Application Description Language (GADL), and later, compiles job descriptions for individual experiments from it. *GrapBench* dynamically controls the total number of benchmarking experiments for individual applications w.r.t. different problem-sizes. In layer 2, these experiments are executed on available Grid-sites provided by resource manager [24]. *GrapBench* considers the Grid-site at both micro level (the individual Grid nodes) and macro level (the Grid-site) taking machine-size as a variable in benchmark measurements. Such application benchmarks easily incorporate the variations in application’s performance associated to different problem-sizes and machine-sizes. Layer 3 computes the benchmarks and stores the results in benchmarks repository. The layer 4 is an analysis and visualization layer making the benchmarks and related analysis available to different clients through GUI and GT4 service interfaces.

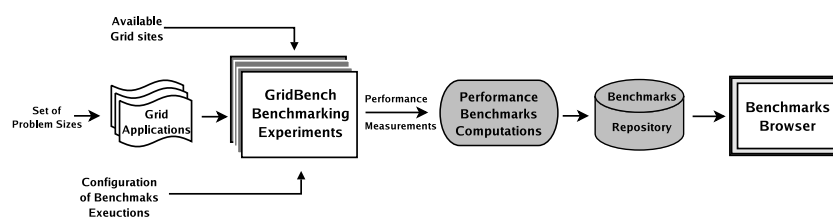


Figure 2: Process of benchmarks generation

4 Benchmarking Measurements

Controlling the number of benchmarking measurements is of key importance in the efficiency of the whole computation process. For the efficiency of computation process we focus to reduce the total number of benchmarking experiments and maximize the utility of benchmarking results.

Our methodology of reducing the number of benchmarking experiments is to enable sharing of the benchmarking information across the Grid-sites. The information sharing mechanism is explained in the Section 5. The performance

comparisons (e.g. performance ratios) of different Grid-sites are different for different applications. Moreover, these also vary for different problem sizes and machine-sizes. For performance prediction and Grid-sites' comparisons while considering variations in the problem-size, we make a full factorial design of benchmarking experiments on the Grid and one benchmarking experiment for one fixed problem-size (called base problem-size) on each of other non-identical Grid-sites. We select the smallest problem-size as the base problem-size. Later, at the time of computation of benchmarks from these experimental results, the process of normalization (see Section 5) helps in completing the benchmarks computation for all the Grid-sites. The benchmarks for the base problem-size are used to share information across the Grid-sites. For scalability analysis and prediction, one benchmark experiment for each of different machine-sizes is also made.

The total number of experiments for an application A with p problem-sizes on a Grid with Grid-size of n , with m different machine-sizes is $m \times n \times p$. One of the distinctions of our work is we can greatly reduce the number of benchmarks experiments through our strategy from $m \times n \times p$ to $m + n + p - 2$ ($n + p - 1$ for execution time predictions and $m - 1$ for scalability predictions). Later, we employ prediction mechanism to serve performance values for the problem-size and machine-size combinations that were not effectively measured in the experiments.

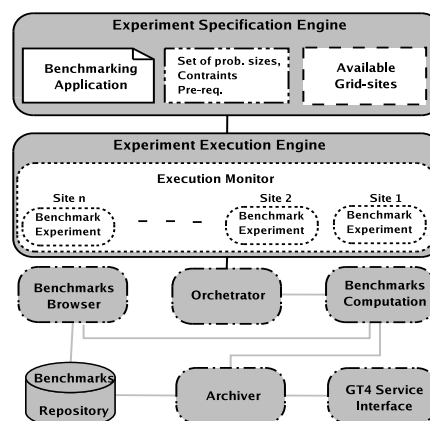
We argue that this reduction in the number of performed benchmarks is a reasonable trade-off between duration of the benchmarking process and accuracy. In Section 5, we show experimentally that predictions based on our approach are with in 90% accuracy. A similar or better accuracy can be achieved with either more benchmarks, or by using analytical modeling techniques. However, both these alternatives are time-consuming. In addition, analytical modeling requires a separate model and expert knowledge for each new type of application. With current grid environments hosting hundreds to thousands of different applications¹, analytical modeling for individual application's performance and scalability (which requires manual efforts) is impractical, whereas, benchmarking requires only one generic setup.

4.1 Grap Benchmarking experiments with AutoGrap

To facilitate Grid administrators, application developers and end users, for comprehensive and adaptable administration and management of benchmarking experiments we have AutoGrap. The architecture of AutoGrap is presented in Figure 3. The main components of the architecture are:

- Experiments Specifications Engine (ESE) (including an RSL/JDL compiler that converts XML specifications of application to job descriptions)
- Experiment Execution Engine
- Monitoring Component
- Orchestrator
- Benchmarks Computation Component
- Archive Component
- Benchmarks Visualization and Analysis Browser
- Information Service Component (publishes results to other services)

First, a small compiler in ESE parses the application specifications written in platform-independent XML based Grid Application Description Language (GADL) and produces the job descriptions from it in a JDL. Later, these job descriptions are add resource specifications, on which these jobs are to be launched, to produce final jobs used for launching the benchmarks experiments. In our present prototype implementations we support the RSL language of Globus [2]. The benchmark application is the actual application's executable, to be benchmarked. The Experiment Execution Engine executes benchmarks experiments designed by ESE on the Grid-sites available from our Resource Manager. The monitoring component watches the execution of the benchmarking experiments and alerts the Orchestrator component to collect the data and coordinate the start-up of the benchmarks computation component to compute



¹The Grid Workload Archive, available online at <http://gwa.ewi.tudelft.nl>, hosts grid workload information that AutoGrap can use to schedule.

the benchmarks. The Archive Component stores this information in the benchmarks repository for future use. The benchmarks Browser publishes the Grap Benchmarks on a GUI, and Information Service Component is an interface to other services for Grap Benchmarks.

4.2 Experiment specification engine

To describe application specifications we introduce Grid Application Description Language (GADL). A GADL definition specifies the application to be executed, its exact paths from resource manager, the problem-size ranges and pre-requisites of execution (some executions before the actual execution or setting of environmental variables etc.) [3, 20], if any. Every instance of GADL is described by:

- Application name with a set of problems sizes given as range (startVal:stepSize:endVal) which are described by name and value, e.g.


```
<application name="Wien2k" />
<parameter>
  <name="k-points" value=5.0:0.1:9.0>
</parameter>
```
- Resource manager [24] URI, to get the available Grid-sites and location of the executables on them.


```
<resourcemanager>
  <location path="http://karwendel.dps.uibk.ac.at:40105/wsrf/
  services/GlareService"/>
</resourcemanager/>
```
- A set of pre-requisites, comprising of the components which must be finished before the execution (of some components of) the application.


```
<prerequisites>
  <location path="http://dps.uibk.ac.at:/home/farrukh/pre-reqs"/>
</prerequisites>
```
- A set of input files required for execution of the application


```
<inputfile>
  <location path="http://dps.uibk.ac.at:/home/farrukh/input.tar"/>
</inputfile/>
```
- An executable to change the problem-size in some input files (used by ESE)


```
<probsizechange>
  <location path="http://dps.uibk.ac.at:/home/farrukh/
  changeProbSize"/>
</probsizechange/>
```

4.3 Experiment execution engine

The AutoGrap exploits *opportunistic load balancing* [22] for scheduling benchmarking experiments in the Grid. The algorithm for automatically making benchmarking experiments is shown in Algorithm 4.3. At the start, one job is submitted to each of the available Grid-sites. The next job is submitted after the completion of the previous submitted job, until all the jobs are finished from the full factorial design of experiments (Algorithm line 5 to line 13). In the next step, one experiment, for one fixed problem-size (common for all, called base problem-size), is made on each of non-identical Grid-sites (Algorithm line 14 to line 17). The benchmarks for the common problem-size are later used in the normalization 5. We categorize two Grid-sites to be identical if they have same CPU architecture, CPU speed, and memory. All benchmarking experiments are made when the Grid-site is found to be free, with the help of NWS [29]. The execution times of these experiments are archived and later used by Benchmarks Computation Component to calculate the benchmarks.

Algorithm 1 Scheduling Benchmarking Experiments

```
1: makeBenchmarkExperiments()
2: Input:  $A : A = \{\alpha, \beta, \gamma, \dots\}$  {Applications to be benchmarked},  $\lambda : \lambda = \{m_1, m_2, m_3, \dots, m_n\}$  {Set of problem-sizes for each application in A},
    $S : S = \{s_1, s_2, s_3, \dots, s_n\}$  {Set of Grid-sites}
3: Output:  $\omega$  - the execution time result set
4:  $Jobs = getJobDescriptions(A, \lambda, S)$ ;
5: while  $Jobs \neq \phi$  do
6:   if  $\exists s \in S \mid available(s)$  (when no job is running on  $s$ ) then
7:      $j = getNextJob(Jobs)$ ;
8:      $\omega' := getExeTime(s, j)$ ;
9:      $\omega := \omega \cup \omega'$ ;
10:  else
11:     $wait()$ ;
12:  end if
13: end while
14: for  $\forall s \in S$  do
15:    $\omega' := getExeTime(s, m_1)$ ;
16:    $\omega := \omega \cup \omega'$ ;
17: end for
18: return  $\omega$ ;
```

4.4 Additional benchmarking considerations

Sometimes the background load, that is, the applications run by other users, severely affects the performance of some (or even all) the applications in the system. This happens mostly when several applications contend for the same network or processor shares, or when resource utilization is very high and the resource manager is ineffective [6]. However, our benchmarking procedure does not take into account the background load, at least for the moment. The reason is threefold. First, our goal is to quantify the best achievable performance of an application on a grid platform, that is, without the contention generated by additional users. Work in [6] helps quantifying the ratio between the maximum achievable performance and the performance achieved in practice. Second, work in hotspot or symbiotic scheduling [28], helps scheduling applications with overlapping resource requirements such that the overlap is minimized. Third, while mechanisms for ensuring the background load on the resources have been proposed, e.g., in [16], a better understanding of the structure and of the impact of the background load is needed. We plan to investigate aspects of this problem in future work.

5 Computation of Benchmarks

The Grap Benchmarks are computed by normalizing the execution times. The execution times are normalized by dividing all the execution times (for different problem-sizes) with the execution time for a base problem-size selected by default as the largest problem-size (to take in effect of inter process communication) in the set of problem-sizes specified by the user. The normalization mechanism not only makes the performance of different machines comparable but also provides a basis for translating different performance values across different Grid-sites. The normalization of values is based on the observation that for many applications, and in particular for all the applications used in our experimental work, the normalized execution times for different problem-sizes and machine-sizes are the same on all the Grid-sites, with 90% accuracy. This allows cross-platform interoperability of Grap Benchmarks. For example, the normalized execution on a Grid-site s for a certain problem-size and machine-size will be equal to that of an other Grid-site t .

$$Normalized\ Exe.\ Time(s) = Normalized\ Exe.\ Time(t)$$

On the basis of our experimental observations, we assume a simple application model under which the rate of change of execution time for a problem-size q on Grid-site s , $T_q(s)$, with respect to execution time for a problem-size r on the a Grid-site s is similar to that of on Grid-site t [17], i.e.

$$\frac{\Delta T_q(s)}{\Delta T_r(s)} = \frac{\Delta T_q(t)}{\Delta T_r(t)}, \Delta : \text{rate of change}$$

For our set of grid applications, this assumption was over 90% accurate, and this is shown in the Figure 4(a). Similarly, we normalize the execution times when machine size is also taken as a parameter, to compute benchmarks incorporating the variations in machine size. This normalization is also based on a similar model based on our experimental

observations. Under this model for CPU-intensive applications, we assume the rate of change in execution time of an application across different problem-sizes is also preserved for different machine-sizes, as shown in Figure 4(b). i.e.

$$\frac{\Delta T_{q,m}(s)}{\Delta T_{q,n}(s)} = \frac{\Delta T_{r,m}(s)}{\Delta T_{r,n}(s)}$$

Moreover, rate of change in performance behavior across different machine-sizes is also preserved for different problems-sizes. i.e.

$$\frac{\Delta T_{q,m}(s)}{\Delta T_{r,m}(s)} = \frac{\Delta T_{q,n}(s)}{\Delta T_{r,n}(s)}$$

We also found an accuracy of more than 90% on this model, and this behavior is also shown in the Figure 4(b).

6 Grap Benchmarks Utilization

Grap Benchmarks are computed from results of benchmarks experiments and archived for future references. This is done in a manner that facilitates the comparisons between the benchmarks for different machines, problem-sizes and machine-sizes, along with the performance and scalability predictions. Benchmarks can be browsed through a GUI (see Figure 8(a)) for application performance and scalability analysis for different problem-sizes on different Grid-sites. In this section we explain how Grap Benchmarks are used for performance & scalability predictions and Grid-site comparisons.

6.1 Performance and scalability predictions

The first key use of Grap Benchmarks is application performance and scalability predictions for Grid users, as well as Grid services like meta-schedulers and performance analysis service. Grap Benchmarks can facilitate good engineering practices by allowing alternative implementations to be compared quantitatively (e.g. see Figure 7(a)). Performance of an application can be predicted, for any problem-size p on any Grid-site s from another Grid-site t (for which execution time for problem-size p exists) from Grap Benchmarks using the phenomenon of normalization as: if $T_q(s)$ represents the execution time of an application, for a problem-size q , on a Grid-site s , then;

$$T_q(s) = \frac{T_q(t)}{T_r(t)} \times T_r(s)$$

Similarly, for scalability analysis and prediction, (taking machine-size as a parameter) the performance of the parallel applications for different number of CPUs can be predicted from Grap Benchmarks as: If $T_{q,m}(s)$ represents execution time of an application for problem-size q on a Grid-site s for a machine-size m , then;

$$T_{q,m}(s) = \frac{T_{r,m}(s)}{T_{r,n}(s)} \times T_{q,n}(s)$$

For execution time and scalability predictions, normalization is done based on execution time for the closest set of parameters(problem-size and machine size). At the start, it is made based on the only common set of parameters in the benchmark repository and later, if some other performance values are available (after adding some experimental values from real runs), calculated based on the closer performance value, as it increases accuracy in the cross platform performance and scalability predictions. For our prediction results we obtained a minimum accuracy of 90% from our proposed number of experiments.

$$\frac{T_{q,m}(s)}{T_{q,n}(s)} = \frac{T_{r,m}(s)}{T_{r,n}(s)} | n \rightarrow m, q \rightarrow r$$

The benchmarks, performance and scalability predictions can be obtained through Information Service Component which is a GT4 service interface, and can also be browsed through Benchmarks Browser GUI.

6.2 Grid-site comparisons

The quantitative performance comparisons of different Grid-sites is different for individual applications. This is because of their different underlying architectures and different performance requirements of Grid applications. This is demonstrated with quantitative comparisons of two Grid-sites *agrid1* and *altix1.uibk* for three real world applications in the Figure 7(b). In addition, the performance comparisons for the same Grid-sites are also different for different problem sizes and machine sizes. This behavior of Grid-applications is presented with performance comparisons of two Grid-sites *altix1* and *agrid1*, in terms of execution time ratios for different problems sizes in Figure 8(b).

The first key use of Grap Benchmarks is support for quantitative comparisons of different Grid-sites. Quantitative comparisons of different Grid-sites help real time schedulers for mapping jobs to different Grid-sites, and resource brokers for resource selections from the available pool of resources. Furthermore, these comparisons provide application developers with information about the systems capabilities in terms of application performance, so that they can develop and tune their applications for high-quality implementations.

Design of Grap Benchmarks helps facilitating the comparisons of applications' performance for different values of problem-sizes and machine-sizes on different Grid-sites, as the second key use. This can guide the Grid-site selection policies by the real time schedulers, resource brokers and different Grid users. The comparison of different Grid-sites for a application Wien2k is shown in the Figure 5(a).

7 Results and Analysis

We have conducted our experiments from the prototype implementation of our system on Austrian Grid. The testbed is described in Table 1. In the test environment, the 8 sites employ 5 cluster sizes, 5 structural and communication architectures, 6 processor types, and 6 memory sizes. This ensures that the tested is heterogeneous. The experiments were conducted for three real world applications Wien2k [8], MeteoAG [21] and Invmod [25].

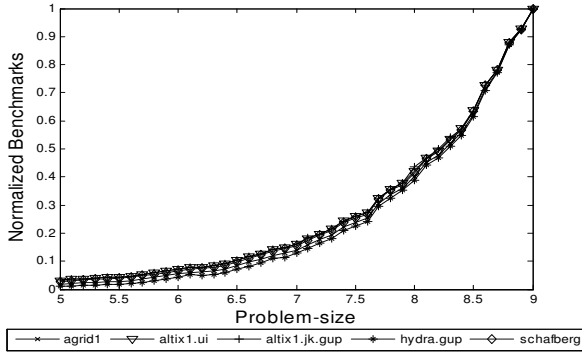
Wien2k application allows performing electronic structure calculations of solids using density functional theory based on the full-potential augmented plane-wave ((L)APW) and local orbital (lo) method. MeteoAG produces meteorological simulations of precipitation fields of heavy precipitation cases over the western part of Austria with RAMS, at a spatially and temporally fine granularity, in order to resolve most alpine watersheds and thunderstorms. Invmod application helps in studying the effects of climatic changes on the water balance through water flow and balance simulations, in order to obtain improved discharge estimates for extreme floods.

The Grap Benchmarks for Wien2k on different Grid-sites of the Austrian Grid are shown in Figure 4(a). Total 45 benchmark experiments were made for 41 different problem-sizes on 5 different Grid-sites. By one experiment here we mean average of multiple repetitions to reduce the anomalies in the computations. In our presented work we repeated each computation for 5 times. For Wien2k, the execution time for problem-size 9.0 is used as base performance value for normalization. The similar benchmarks curves (for different values of problem-size) on different machines show the realization of normalized performance behavior of the Grid Benchmarks across heterogeneous platforms.

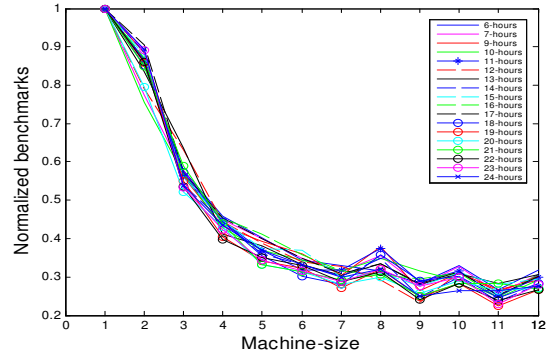
To give a glimpse of the variability in the quantitative comparisons of different Grid-sites for different applications, we present our experimental results in Figure 7(b). As shown in the figure, the comparison of two different Grid-sites *agrid1* and *altix1.uibk* yielded different ratios (*altix1.uibk* : *agrid1*) of execution times for three different applica-

Table 1: The Austrian Grid test bed sites.

Site Name	Architecture	CPUs	Processor Arch.	RAM[MB]	Location
altix1.jku	ccNUMA, SGI Altix 3000	16	Itanium 2, 1.6	1400	Linz
altix1.uibk	ccNUMA, SGI Altix 350	16	Itanium 2, 1.6	1600	Innsbruck
schafberg	ccNUMA, SGI Altix 350	16	Itanium 2, 1.6	1400	Salzburg
agrid1	NOW Fast Ethernet	20	Pentium 4, 1.8	1800	Innsbruck
hydra	COW Fast Ethernet	16	AMD Athelon 2.0	1600	Linz
hcma	NOW Fast Ethernet	206	AMD Opteron 2.2	4000	Innsbruck
zid-cc	NOW Fast Ethernet	22	Intel Xeon 2.2	2000	Innsbruck
karwendel	COW InfiniBand	80	AMD Opteron 2.6	16000	Innsbruck
Total		388			

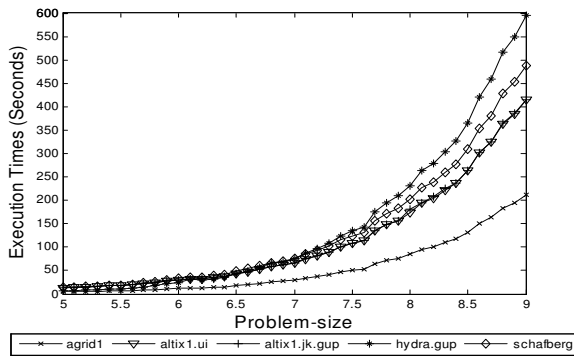


(a) Wien2k, 41 problem-sizes, 5 Grid-sites

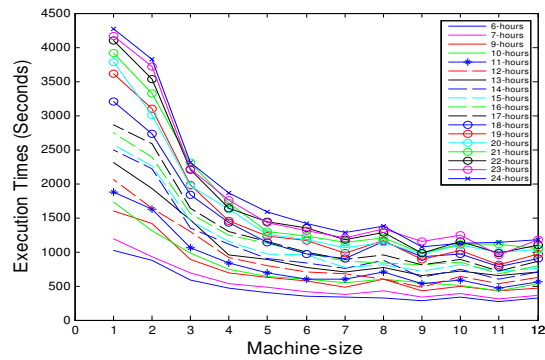


(b) MeteoAG on *zid-cc*, 19 problem-sizes, 12 machine sizes

Figure 4: (a) Normalized benchmarks, similar curves for different Grid-sites(a), and different machine-sizes(b) exhibit the realization of normalized behavior



(a) Wien2k, Grid sites comparisons, performance benchmarks for 41 different problem-sizes, 5 Grid sites



(b) MeteoAG on *zid-cc*, scalability comparisons for different machine sizes, performance benchmarks for 19 problem-sizes, 12 machine sizes

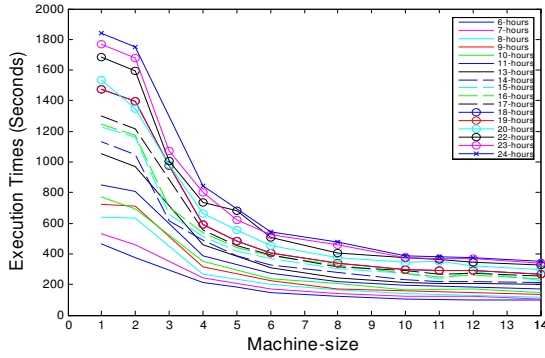
Figure 5: Grid-sites and scalability comparison for different applications

tions. For Wien2k this ratio is 2.37, for Invmod 10.37 and for MeteoAG 1.71. It is noteworthy that these ratios are irrespective of the total execution times on these Grid-sites. This is the reason that why benchmarks for individual resources (CPU, memory etc.) do not suffice for application performance and scalability predictions. Furthermore, considering one application, the comparison of execution times on Grid-sites yields different ratios for different problem-sizes. This performance behavior of Grid applications urged us to make a full factorial design of experiments on the Grid, rather than modeling individual applications analytically which is complex and in-efficient. The execution time ratios of two Grid-sites *altix1.uibk* and *agrid1* for 41 different problem-sizes are shown in Figure 8(b).

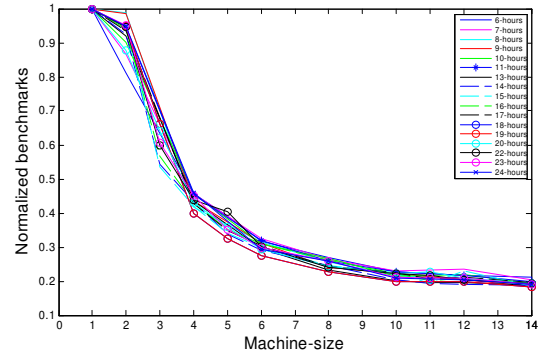
Performance and scalability benchmarks for different number of CPUs, for an other application MeteoAG are shown for two different Grid-sites *zid-cc* and *hcma* in Figures 5(b) and 6(a) respectively. Total 30 benchmarking experiments were made for 19 different problem-sizes and 12 different machine sizes on *zid-cc*. Total 32 benchmarking experiments were made for 19 different problem-sizes and 14 different machine sizes on *hcma*. In these experiments we have used a machine-size of 1 for normalization. The identical scalability curves demonstrate the realization of normalized performance behavior of Grid Benchmarks with respect to problem-size and machine-size on one platform.

A comparison of different Grid-sites, based on Grap Benchmarks, for Wien2k is shown in Figure 5(a). The scalability comparison for MeteoAG for different problem-sizes, on two different platforms (*zid-cc*(32-bits) and *hcma*(64-bits)) is shown in Figures 5(b) and 6(a) respectively. A comparison of two different version of Wien2k (32-bits version and 64-bits version) is presented in Figure 7(a) on *karwendel*. These graphs were generated from Grap Benchmarks, when only one benchmark measurement of 64-bits version was available.

We observed a maximum average variation of 10% from actual values (obtained from real runs) in our performance and scalability predictions which means a 90% accuracy in our predictions where the maximum standard deviation of 2% was observed.

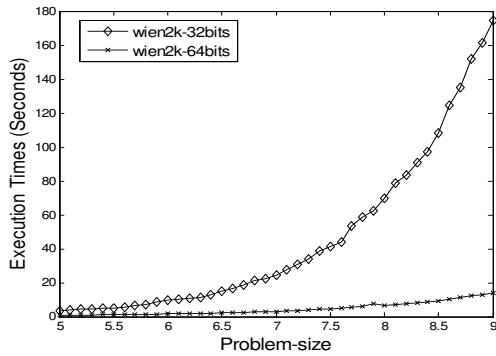


(a) MeteoAG on *hcma*, Scalability comparison for different machine-sizes for 19 different problem-sizes and 14 machine sizes

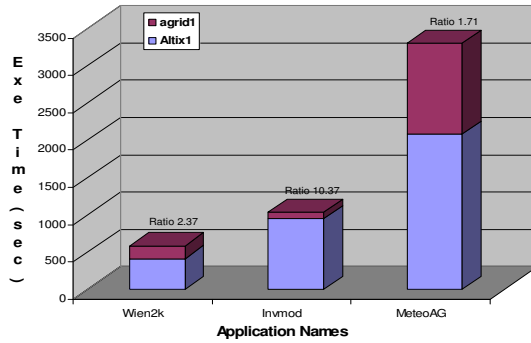


(b) MeteoAG on *hcma*, Normalized benchmarks for 19 different problem-sizes and 14 machine sizes, similar curves show the realization of normalized behavior

Figure 6: MeteoAG Grap Benchmarks on *hcma*



(a) Wien2k on *karwendel*, version comparisons for different problem-sizes from benchmarks, 32-bit and 64-bits



(b) Quantitative performance comparison of *altix1* and *agrid1* for 3 different applications, different ratios show different comparisons

Figure 7: Different comparisons, based on Grap Benchmarks

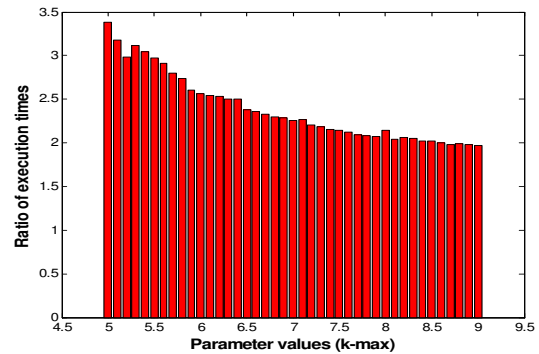
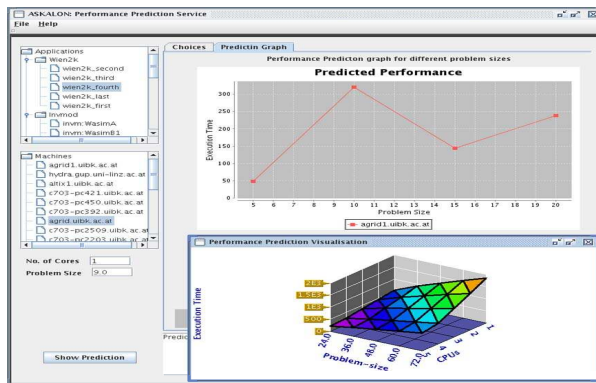
8 Related Work

There have been several good efforts at benchmarking individual Grid resources like [4, 7, 18, 1]. The discussion presented in [10] shows that the configuration, administration and analysis of NGB requires an extensive manual effort, like other benchmarks. Moreover, these benchmarks lack some integration mechanism needed to make meaningful inferences about the performance of different Grid applications from these benchmarks.

A couple of good comprehensive tools like [27] are also available for benchmarking a wide range of Grid resources. These provide easy archiving and publishing the results. Likewise, GrenchMark [15] is a framework for analyzing, testing, and comparing grid settings. Its main focus is the generation and submission of synthetic grid workloads. By contrast, our work focuses on single application benchmarks, which are extensively supported.

Individual benchmarks have been successfully used for resource allocation [19, 5] and application scheduling [14]. Work in [19] presents good work for resource selection, by building models from resource performance benchmarks and application performance details. Authors in [5] present resource filter, resource ranker and resource MakeMatch on the basis of benchmarks and user provided information. Though this work provides good accuracy, it requires much user intervention during the whole process. Moreover, these benchmarks do not support cross-platform performance translations of different Grid applications while considering variations in problem-sizes.

A similar work has been presented by Dikaiakos et. al. in [26]. The authors present a tool for resource selection for different applications while considering variations in performance due to different machine-sizes. Importance of application specific benchmarks is also described by [23]. In this work authors present three different methodologies to benchmarks Grid application by modeling application and Grid-site, and require much manual intervention.



(a) Graphical user interface, for performance and scalability analysis

(b) Quantitative Grid-sites comparison for different problem sizes, execution times ratios for *altix1.uibk* and *agrid1* (*altix1.uibk* : *agrid1*) for 41 different problem sizes

Figure 8:

The distinction of our work is that we focus on controlling and specifying the total number of experiments needed for benchmarking process and our proposed benchmarks are flexible regarding variations in machine-size as well as problem-sizes (required for real time scheduling and application performance prediction). Moreover, we support a semi-automatic benchmarking process. The cross platform inter-operability of our benchmarks allows trade-off analysis and translation of performance information between different platforms.

9 Conclusion and Future Work

Application benchmarks provide a concrete base for application's performance analysis and predictions, incorporating variations in the problem-sizes and machine-sizes on different platforms and for real quantitative comparison of different Grid-sites for individual applications. Grap Benchmarks are representative of performance of Grid applications on different Grid-sites. Computed from real execution of applications these incorporate the individual effects of different Grid resources specific to applications. Their computation, execution, performance measurements storage and management are affordable in terms of cost, such as management of the benchmarking process, execution time and storage. Performance metrics derived from the benchmarking experiments can be easily associated with the structure of the corresponding benchmarks. Grap Benchmarks are easy to compute and use in contradiction to existing benchmarking tools and techniques for different Grid resources. Experiments conducted according to our proposed strategy make the benchmarks flexible regarding problem-size and thus allow problem-size and machine-size flexible comparisons of different Grid-sites. Simple metrics of benchmarks make them directly usable and understandable to different Grid users and services.

We benchmark applications' performance only from execution times. The reason for this is general property of benchmarks that these should be repeatable and queue wait times are not normally repeatable. Grap Benchmarks can also help in designing trade-off analysis, and to exchange information between infrastructure developers and Grid applications writers. We are enhancing our present work towards application benchmarking at the level of Grid constellations comprised of multiple sites which constitute the computing platform of a Virtual Organization. We also plan to incorporate application throughput information for performance transformation across the platforms.

References

- [1] <http://www.netlib.org/parkbench.html/>.
- [2] Globus toolkit. <http://www.globus.org/>.
- [3] Grid benchmarking group, global grid forum. <http://forge.gridforum.org/projects/gb-rg> (Accessed 2005).
- [4] Spec benchmarks. <http://www.spec.org/>.

- [5] Enis Afgan, Vijay Velusamy, and Purushotham V. Bangalore. Grid resource broker using application benchmarking. In *EGC*, pages 691–701, 2005.
- [6] Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Amin Vahdat, Lok T. Liu, Thomas E. Anderson, and David A. Patterson. The interaction of parallel and sequential workloads on a network of workstations. In *SIGMETRICS*, pages 267–278, 1995.
- [7] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, 1991.
- [8] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz. *WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties*. Institute of Physical and Theoretical Chemistry, TU Vienna, 2001.
- [9] M. De Roure, D. Surridge. Interoperability challenges in grid for industrial applications. In *In Proceedings of GGF9 Semantic Grid Workshop, Chicago IL, USA, May 2003*.
- [10] Rob F. Van der Wijngaart and Michael A. Frumkin. Evaluating the information power grid using the nas grid benchmarks. In *IPDPS*, 2004.
- [11] Marios D. Dikaiakos. Grid benchmarking: vision, challenges, and current status: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(1):89–105, 2007.
- [12] T. Fahringer and Radu Prodan et. al. ASKALON: A Grid Application Development and Computing Environment. In *6th International Workshop on Grid Computing (Grid 2005)*, Seattle, USA, Nov 2005.
- [13] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA, 2004.
- [14] Elisa Heymann, Alvaro Fernández, Miquel A. Senar, and José Salt. The eu-crossgrid approach for grid application scheduling. In *European Across Grids Conference*, pages 17–24, 2003.
- [15] Alexandru Iosup and Dick H. J. Epema. Grenchmark: A framework for analyzing, testing, and comparing grids. In *CCGRID*, pages 313–320, 2006.
- [16] Hashim H. Mohamed and Dick H. J. Epema. Experiences with the koala co-allocating scheduler in multiclusters. In *CCGRID*, pages 784–791, 2005.
- [17] Farrukh Nadeem, Muhammad Murtaza Yousaf, Radu Prodan, and Thomas Fahringer. Soft benchmarks-based application performance prediction using a minimum training set. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 71, Amsterdam, Netherlands, December 2006.
- [18] Netlib. <http://www.netlib.org/linpack/>.
- [19] G. R. Nudd and S. A. Jarvis. Performance-based middleware for grid computing: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(2-4):215–234, 2005.
- [20] Xiao Dong Wang Rob Allan and Andy Richards. Xml schema for hpc applications and resources. Grid Technology Group. Daresbury Laboratory (2002).
- [21] Jun Qin Schueller Felix and Farrukh Nadeem. Performance, Scalability and Quality of the Meteorological Grid Workflow MeteoAG. In *2nd Austrian Grid Symposium, Innsbruck, Austria, Sep 2006*.
- [22] Uwe Schwiegelshohn and Ramin Yahyapour. Fairness in parallel job scheduling. *Journal of Scheduling*, 3(5):297–320, 2000.
- [23] Margo I. Seltzer, David Krinsky, Keith A. Smith, and Xiaolan Zhang. The case for application-specific benchmarking. In *Workshop on Hot Topics in Operating Systems*, pages 102–, 1999.

- [24] Mumtaz Siddiqui, Alex Villazon, Jurgen Hofer, and Thomas Fahringer. Glare: A grid activity registration, deployment and provisioning framework. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 52, Washington, DC, USA, 2005.
- [25] Dieter Theiner and Marek Wieczorek. Reduction of calibration time of distributed hydrological models by use of grid computing and nonlinear optimisation algorithms. In *Proceedings of the 7th International Conference on Hydroinformatics (HIC 2006)*, Sep. 2006.
- [26] A. Tiramo-Ramos, G. Tsouloupas, Marios Dikaiakos, and Peter M. A. Sloot. Grid resource selection by application benchmarking: a computational haemodynamics case study. In *Computational Science - ICCS 2005, 5th International Conference Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part I.*, volume 3514, pages 534–543, Atlanta, Georgia, USA, May 2005.
- [27] George Tsouloupas and Marios D. Dikaiakos. Gridbench: A tool for benchmarking grids. *Grid Computing*, page 60, 2003.
- [28] Jonathan Weinberg and Allan Snavely. Symbiotic space-sharing on sdsc's datastar system. In *JSSPP*, pages 192–209, 2006.
- [29] Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.
- [30] Globus Alliance WSRF. Web services resource framework. <http://www.globus.org/wsrf>.