



Project no. FP6-004265

## CoreGRID

European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies

Network of Excellence

GRID-based Systems for solving complex problems

### D.SA.02 – Proposal of Architectural Components for Scalable, Adaptive and Dependable GRIDs

Due date of deliverable: December 31, 2005

Actual submission date: January 29, 2006

Actual submission date (revised): April 3rd, 2007

Start date of project: 1 September 2004

Duration: 48 months

Organisation name of lead contractor for this deliverable:

Zuse Institute Berlin (ZIB)

Revision draft

<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>PU</b>

**Keyword List:** Grid and P2P systems architecture, dependability, scalability, adaptability, Peer-to-Peer systems, desktop Grids

## 1 Introduction

In this document we give an overview of the major technical results of the Institute on System Architecture for the deliverable D.SA.02, the "Proposal of Architectural Components for Scalable, Adaptive and Dependable GRIDs". The results cover all of these three areas and have been presented originally and in full detail in scientific papers of the Institute members.

The summarized results reflect well the joint scientific achievements of the partners related to the aspects of Grid system architecture as defined by the Joint Program of Activities version 1 and 2 and the Roadmap version 1 of WP4. This research work has been carried out initially under the umbrellas of WP4 Tasks 4.3, 4.4 and 4.5 in JPA01. In JPA02, the additional structure of dynamic Research Groups has been introduced to allow more focus yet greater dynamics of partners' cooperation. The architectural aspects targeted in the tasks/Research Groups include scalability (in connection to P2P systems), dependability and adaptability. Also, some Research Groups link the topic of Desktop Grids and their properties with research concerning dependability and scalability. All these research fields cover important issues in the wide domain of Grid architectures, and the methods and solutions proposed by the WP4 are thus regarded as components or "building blocks" for architectures of future Grids.

The last part of this document present a comparison between the topics covered by the contributions of the WP4 researchers to the deliverable D.SA.02 versus the research fields discussed in roadmap version 1. The comparison helps to obtain of a quick view of areas where the roadmap possibly must be adjusted, or where acquiring more expertise within WP4 is necessary.

In order to contribute to the dissemination activities, the results of the Institute have been presented in papers and presentations of the authors in two CoreGRID workshops. The first one was the *CoreGRID Integration Workshop* in Pisa (November 2005), and the second, more focused on System Architecture, the *2nd CoreGRID Workshop on GRID and Peer to Peer Systems Architecture*. The latter took place at LIAFA, Universit Paris 7, Paris, France on 16 and 17 January 2006. The call for papers of the workshop was open to all scientists (also external to CoreGRID), however the majority of the submitted papers (13 out of 14) come from researchers related to Institute on System Architecture (WP4).

## 2 Scalability and Peer-to-Peer Systems

The work of the Institute on scalability and peer-to-peer systems for Grids covers a wide spectrum, which include the research topics about resource discovery, data integration and algorithm and so on. As for resource discovery, a research about a survey paper and a research with proposed architecture for resource discovery using range queries have been presented. Data integration research focuses on a hybrid of unstructured and structured peer-to-peer systems. And

the research on algorithm of Peer-to-Peer systems present an algorithm for reducing duplicate messages in unstructured peer-to-peer systems and a paper that gives a general approach for building self-managing distributed systems.

## **2.1 Resource management**

### **2.1.1 Survey of resource discovery**

The research of the institute for the deliverable D.SA.02 concerning resource discovery for Grids using Peer-to-Peer techniques have been presented in [18]. At present, there are a large number of proposals to achieve resource discovery for Grids, which the article classifies them according to whether they are based on unstructured or structured peer-to-peer systems or on semantic information. In the process of the presentation, the paper first reviews the most promising Grid systems that use P2P techniques to facilitate resource discovery in order to perform a qualitative comparison of the existing approaches and to draw conclusions about their advantages and weaknesses. Then discusses the system standardization efforts that will lead to full-scale deployment and future research directions in the field. In this discuss, the authors first review four main grid environments that provide resource discovery capabilities, Globus, Condor, UNICORE, and LCG/EGEE, on which the resource discovery are based on centralized or hierarchical client/server models. For example, in Globus Toolkit 4, resource discovery is addressed by the MDS (Monitoring and Discovery Service), which provides a hierarchical organization based on Index Services, which are Grid services that hold information about a set of other Grid services registered to it. After this, they discuss the resource discovery on peer-to-peer systems, which they have classified as unstructured P2P systems, structured P2P systems and hybrid systems, and make a comparison of their scalability, robustness, periodic update and ranged queries. Different from these traditional desktop based P2P systems, grid based P2P systems are the newer techniques that combine P2P approach and grid system. To introduce these newer techniques, the authors review some recently proposed systems that adopt the P2P approach to the P2P approach for grid resource discovery, both unstructured system and structured system, which several approaches which are based on structured peer-to-peer systems using query algorithms have developed. For example, the protocols for Chord, CAN, and Pastry have been extended to handle range queries.

In the last part of the paper, the authors summarize the relative advantages and disadvantages of the unstructured versus structured techniques. Unstructured systems are more suited to highly dynamic Grids whereas structured systems offer better performance. The article ends by proposing that more work be done in semantic resource discovery, a area that is considered by the authors to be especially promising, with which can improve the precision of a discovery service.

### 2.1.2 Resource management

The research results of the institute in WP4 about Resource management to Peer-to-Peer systems have been presented in [5]. In this research, the use of currently widely used Peer-to-Peer systems to improve drawbacks caused by centralized approach has been used and the architecture of a new system to handle efficiently the management of resources for grid, and a new peer to peer system to fulfill those lacks has been proposed. To clearly present the research results, the author firstly reviews a most popular used system for resource handle Globus, then introduce file share system, peer-to-peer systems both classical peer to peer systems and more new peer to peer systems that use for grid system. After this, they try to present a design of an efficient system to discover and request resources for grids which can fast change state while well distribute the workload over the nodes. Using a range query algorithm on a structured peer-to-peer system to request the system discovers resources according to complex queries and handles a typical request that is a range request of the form "request from L to U nodes" where L is a lower bound and U is an upper bound. This request is handled by an augmented tree structure of nodes, where each node handles a particular range of L to U. The tree is augmented with short-cut links to improve performance. To implement this system, the authors propose a two layer architecture in which the peer-to-peer resources are classified as stable ones and dynamic ones. Using this architecture they propose PRAG (Peer-to-peer Resources mAnager for Grid), by which users need not care of the cost of the update for stable resources, and only a tradeoff is needed between request and update for dynamic resources, which is very simple and low cost.

## 2.2 Data integration

The research results of the institute for the deliverable D.SA.02 concerning data integration have been presented in [4]. Here the presented architecture is a decentralized one for data integration, called PARIS, which is based on schema mapping. PARIS is a hybrid architecture that combines both unstructured and structured techniques. It models the system as a collection of peers where each peer corresponds to a single data source. Each data source has a local schema and a collection of mappings to foreign schemas. The system supports two kinds of mappings: point-to-point mappings and transitive mappings. In the latter, data sources are related through one or more "mediator schemas" at intermediate peers, through which the user can retrieve data from all the related sources in the system by submitting a single query.

Peers with the same schema are kept connected by using an unstructured, gossip-based protocol. This makes the architecture robust in the face of high churn. In parallel to this, peers also participate in a DHT structure. This permits efficient global searches to be done. Query processing is done by a variation of a flooding algorithm that uses the DHT to reduce the number of superfluous messages (by sending messages to just one representative for each peer group).

PARIS was tested in the PeerSim simulation framework for system sizes up to 100000 peers. Its future work will focus on implementation a full software prototype and deployment it on a test bed such as PlanetLab.

## **2.3 Algorithms and architecture**

### **2.3.1 Reducing duplicate messages in unstructured peer-to-peer systems**

The research of the institute for the deliverable D.SA.02 in WP4 concerning the architecture of Peer-to-Peer systems [13] have presented a resource location algorithm for unstructured peer-to-peer systems that reduces the number of duplicate messages. Typically, an unstructured system uses a flooding algorithm for resource location, but this is extremely wasteful in number of messages. To reduce the number of duplicate messages, the algorithm of this paper runs in two phases. In a short initial warm-up phase, statistics are gathered on the number of duplicate messages for each node. In the second phase, each node decides whether or not to forward messages based on whether a high percentage of duplicates passed through it during the warm-up phase.

Simulation shows that this algorithm can significantly reduce network traffic both in random graphs and small-world graphs. With graphs of 2000 nodes, the number of duplicates was reduced to less than 20% while conserving network coverage above 80%.

### **2.3.2 Self management**

The research [14] presents the results of the institute in WP4 about self management in large-scale distributed Peer-to-Peer systems. The presented architecture is intended to cover the classical definition of self management as the ability of a system to reconfigure itself to handle changes in its environment or requirements without human intervention but according to high-level management policies. The approach combines structured peer-to-peer systems with an advanced component model. The peer-to-peer system already does self management at the lowest level, for topology maintenance. The component model, if it is well chosen, allows the feedback mechanisms to be implemented for doing self management at higher levels. One component can monitor a property of the system, another component calculates a correction based on this monitored value, and a third component can apply the correction to the system. Usually, standard component models such as Enterprise Java Beans and the CORBA Component Model are not expressive enough for this; but a more advanced component model is needed that is reflective and has more control. One such model is the Fractal model developed by INRIA and France Telecom. This approach will be applied to an important application scenario, the multi-tier application, in a European project that will start later this year.

### 3 Dependability

One of the topics of paramount importance in the development of Grid middleware is the impact of faults since their probability of occurrence in a Grid infrastructure and in large-scale distributed system is actually very high. So it is mandatory that Grid middleware should be itself reliable and should provide a comprehensive support for fault-tolerance and adaptivity mechanisms, like failure-detection, fault-diagnosis, checkpointing-recovery, replication, software rejuvenation, component-based reconfiguration, among others.

#### 3.1 Dependability evaluation and Dependability Benchmarking

In a network consisting of several thousands computers, the occurrence of faults is unavoidable. Being able to test the behavior of a distributed program in an environment where users can control the faults (such as the crash of a process) is an important feature that matters in the deployment of reliable programs. One of the techniques to evaluate the effectiveness of those fault-tolerance mechanisms and the reliability level of the Grid middleware is to make use of some fault-injection tools and robustness testers to conduct some experimental assessment of the dependability metrics of the target system.

The research results of the institute for the deliverable D.SA.02 on fault injection techniques have been presented in [16] and [9]. [16] discusses several software fault injection tools and workload generators, which are used to evaluate the dependability of the target systems. The emphasis is on the FAIL-FCI fault-injection software that has been developed in INRIA Grand Large, and a benchmark tool called QUAKE that has been developed in the University of Coimbra. In the research, the researchers make a comparison of FAIL-FCI with some other popular fault injectors for dependability assessment of Grid-based applications and Grid middleware. From the comparison, it clearly shows that FAIL-FCI take advantages over other tools. These advantages come from several reasons as follows: Firstly, the FAIL language is a high level language that allows defining fault scenarios, which can be used to describe state machines that model fault occurrences. And it also can describe the association between these state machines and a computer (or a group of computers) in the network. So it is very convenient for users using this tool. Secondly, FCI is thus a Debugger-based Fault Injector which makes it possible not to have to modify the source code of the tested application, while enabling the possibility of injecting arbitrary faults. Thirdly, the automatically generated fault injection daemons can communicate explicitly according to the user-defined scenario. This allows the injection of faults based either on a global state of the system or on more complex mechanisms involving several machines. In addition, the fully distributed architecture of the FCI daemons makes it scalable, which is very useful in the context of emulating large-scale distributed systems. Last but not least, FCI daemons' two operating modes: a random mode and a deterministic mode allow fault injection based on a probabilistic fault scenario or based on a deterministic

and reproducible fault scenario. Using a debugger to trigger faults also permits to limit the intrusion of the fault injector during the experiment. Beside fault simulation tools, the dependability evaluation standards, benchmarking are also of most importance to assess the dependability of grid systems. On these issues, the authors brief the current state of the art on the dependability evaluation of the distributed systems, and give a detailed discussion on QUAKE tool, a dependability benchmark tool for Grid services. In the discussion, they also present the whole application process of a concrete example on how to use the QUAKE tool, from the experimental setup, to benchmark procedure, workload, faultload, measures and benchmark properties.

The research [9] extends FAIL-FCI (for Fault Injection Language, and FAIL Cluster Implementation, respectively), a software tool that permits to elaborate complex fault scenarios in a simple way, while relieving the user from writing low level code. Interestingly enough, in this fault injection tool, the whole process is driven by a simple unified description language, that is totally independent from the language of the application, so that no code changes or recompilation are needed on the application side.

To validate the versatility of the fault injection method on the distributed systems, the researchers have experimented on XtremWeb environment using FAIL-FCI technique for three kinds of fault injections, that is quantitative fault injection, qualitative fault injection and stress testing on two different kinds of hard settings, Cluster and GridXplorer. Where the XtremWeb is a general purpose platform that can be used for high performance distributed computation, and the Cluster is the cluster with 30 machines with smaller RAMs, the GridXplorer consists of 160 machines with much larger RAMs. Their quantitative fault injection experiments show that some computations actually do not terminate as expected at both settings even though the crash probability of the XtremWeb workers is increased from 10% to 90% with a fix interval. This phenomenon is explained by malfunction of the XtremWeb dispatcher. The collected information about the dispatcher failure shows that in the Cluster setting, the rate of fault appearance does not significantly change the execution time when the failure probability is relatively low, and there occurs some kind of equilibrium, that is the execution time does not vary much when the failure probability enters into some value span, which is also suitable for the GridXplorer setting though it has an impact on performance at different failure probability from the Cluster. Besides these, the authors has also found that in the GridXplorer setting, all tests finish despite the larger set of worker processes, which shows the scalability is probably not the explanation for the dispatcher malfunction in the Cluster case, but resource exhaustion or a problem with thread synchronization does.

The results obtained in the qualitative fault injection experiments show that injecting stop faults induce worse performance than injecting halt faults. And if the workers crash before even starting a computation, the performance is worse than if it crashes after the computation. This behavior is probably explained as a misconception in the XtremWeb dispatcher, that does not expect failures just after the job was sent. Besides it is also found that if a worker crashes after a job is completed the worker notified the controller that the results are

available), then the performance is almost the same as if no faults were injected.

The stress testing fault injection experiments show that in the Cluster setting, all curves about the relations between execution and the probability for a worker start are decreasing, which means that the more workers applied, the faster is the completion of the computation. In the XtremWeb platform there is no problem to handle 30 new workers arriving at the same time while the more workers added simultaneously to the system, the longer time it needs to complete the task in the GridExplorer setting, which means that with 160 workers, many of which arriving simultaneously, the XtremWeb dispatcher is clearly stressed but degrades gracefully.

## 3.2 Adding Fault-tolerance to GRID applications

### 3.2.1 Fault-tolerant MPI in GRIDs

In order to execute without modification Message Passing distributed applications on a computational grid, one has to address many issues. The first to come is how let processes of two different clusters communicate. The research results of the institute in WP4 for the deliverable D.SA.02 concerning fault-tolerant MPI in Grids have been presented in [2]. In the article, the researchers study the performances of relaying techniques that use for communication between different clusters to solve this issue. When using relays, messages and most of the nondeterministic behavior of nodes pass through the relays during the execution. This provides the ability to implement fault tolerance at the relay level using pessimistic message logging techniques. Along this, the authors also evaluate the overhead of the logging and study how relays should be designed and fault tolerance protocols composed to provide a full fault-tolerant Message Passing Interface library for computational grids.

To present their message relaying techniques for computational grids, the authors discuss the whole thought from the architecture, implementation and performance evaluation of the system. As for architecture, they consider the computation grids that are independent high performance clusters bound by Internet links. Clusters consist of a set of homogeneous computers, communicating through a high performance network inside the same cluster, and through gateways using a slower network outside different clusters. To configure the relay, they devise the R-device, which can communicate directly with the relay instead of traditional communication daemon and can provide a fully-functional MPI library, that is blocking emission and reception, probe of messages, initialization and finalization. In the relay mechanism, the pessimistic message logging techniques are adopted based on the relay will not be subject to failure. Because the pessimistic message logging technique is a rollback-based fault tolerant technique which does not enforce synchronization of the checkpoints. It need to log much checkpoint information, both sequence of deterministic phases and nondeterministic actions. To simplify the logging process and decrease the stored information, they use a specificity of the MPICH implementation to reduce the number of information to store, which does not maintain a full event for probes,

but count the number of unsuccessful probes between deliveries.

Because the relay is a key of the architecture, the authors make a special care on the implementations of multiplex many communication channels with the most efficiency and log messages, and have consider two different implementations, single process multiplexing streams and one system per connection. Besides, pool of thread multiplexing streams and specialized threads multiplexing streams are also the issues which the authors focus on.

The results obtained from experiments show that the implementation can have a significant impact on the performance of the relay, for example, for long messages, only the most thread-consuming implementation obtained the medium saturation, while the monothreaded method may provide the same performances for long messages and also provide the best performances for short messages. To maximize performance for relaying technique, the application should be aware of the different networks capacities and node placement, groups and other issues that should be addressed automatically by the grid MPI library. Only coupled with a checkpointing mechanism and a checkpoint scheduler to alleviate the memory consumption of the relays when logging is enabled, remote pessimistic message logging can be of low cost.

### **3.2.2 Fault-tolerant Services in GRIDS**

In order to make sure the efficiency of the enterprise middleware and component platforms, some non-functional aspects such as services must be added to the application functional code and deployed in the component containers. The research of the institute in WP4 for the deliverable D.SA.02 concerning fault-tolerant service in Grids [3] proposes an architecture for defining, configuring, and deploying such Technical Services in a Grid platform. To present the architecture proposed, the authors firstly brief the grid platform in which the technical services are deployed, the grid middleware ProActive a Java library for concurrent, distributed and mobile computing. Then describe the descriptor-based deployment of grid applications in detail from deployment model to resource retrieval techniques which include creation-based deployment and acquisition-based deployment. The reason why they propose to express the non-functional services such as security, fault-tolerance and load-balancing in the deployment descriptors is that the deployment is abstracted to virtual nodes, and what they propose can allow a clear separation between the conceptual architecture using virtual nodes and the physical infrastructure where nodes exist or are created, maintain a clear separation of the roles, and be convenient for experts implementing and providing non-functional services. Along the deployment descriptor, the authors also propose to leverage the definition of deployment non-functional services with the introduction of dynamically applicable technical services. In the technical services, the authors present an implementation for activating and configuring load-balancing at deployment time, which can make sure non-functional requirements be dynamically fulfilled at runtime by adapting the configuration of selected resources. The experimental results show that the computation time decreases with the number of CPUs, and it is hard

to control expected machines with the peer-to-peer aspects of their benchmarks which run on a desktop grid using a different set of machines at each run.

### 3.2.3 Sabotage Tolerance

The success of Grid Computing in open environments like the Internet is dependent on the existence of mechanisms to detect malicious failures and sabotage attempts. Together with those low-level mechanisms it is also required to maintain a trust-management system that permits to distinguish the trustable from the non-trustable participants in a global computation. Without those techniques users with data-critical applications will never be reliable in desktop grids, and will rather prefer to support higher costs to run their computations in a closed and secure cluster environments. The research of the institute in WP4 for the deliverable D.SA.02 concerning sabotage tolerance and trust management in Grids is presented in [15]. For a more detailed discussion, see Section 5.1.

Except the research results mentioned above, the institute has also done other researches in WP4 on the topics related to fault-tolerance techniques [3], [2]. In general, we expect in the future work of the Institute on System Architecture a stronger convergence between the topics of adaptability and self-management, with inclusion of some aspects of dependability.

## 4 Adaptability

Generally, the adaptability of Grid systems has not been treated as a separate research topic at the workshop. However, this research area is strongly connected to the selfmanagement and to some aspects of dependability, and so it entered various contributions at the workshop. The research results of the institute in WP4 deliverable D.SA.02 related to this topic have been presented in [4] and [14].

The research [4] proposes a novel peer-to-peer architecture, PARIS, aimed at processing the unprecedented amount of information in the highly heterogeneous, autonomous and dynamic environments. In the proposed architecture, the combination of decentralized semantic data integration with gossip-based overlay topology management and distributed hash tables provides the required level of flexibility, adaptability and scalability, and still allows to perform rich queries on a number of autonomous data sources. In the presentation, the authors have described the logical model that supports the architecture and show how its original topology is constructed, presented the usage of the system in detail, particularly, the algorithms used to let new peers join the network and to execute queries on top of it. As it is to develop a decentralized network of semantically related schemas that enables the formulation of queries over autonomous, heterogeneous and distributed data sources, the PARIS environment is modeled as a system composed of a number of peers, each bound to a data source. Due to peer schemas are connected to each other through declarative

mappings rules, the mapping mechanism in PARIS is flexible and scalable, and it can be used to capture structural as well as terminological correspondences between schemas. On the architecture, the authors design the topology by keeping the topology management and actual data integration at two distinct levels. This separation of concerns allows users to exploit recent algorithmic advances in the later domain on top of a scalable and robust overlay, which means that the system should support a very large number of peers and be able to cope with the highly dynamic nature of the network. The experimental results show that besides the absolute value of the query span which conforms to the theoretical value expected given the network architecture, the self-stabilization property of the network when nodes are joined is its main advantage. Even when the network grows continuously or in the presence of massive flash crowds, only a few cycles are enough for the span value to stabilize to its nominal value.

The research [14] of the institute in WP4 envisions and presents the implementation of making large-scale distributed applications self managing by combining component models and structured overlay networks. To show and verify the usefulness of the implemented self management mechanism, the authors target multi-tier applications, and in particular consider three-tier applications using a self-managing storage service, and propose a self management architecture based on the computer system approach. Compared with traditional approaches, they extend self management from traditional low-level to high-level which includes such as configuration, deployment, online updating, and evolution, which have been largely ignored so far in structured overlay network research. Together with this, they also extend self management to study component-based abstractions and architectural frameworks for large-scale distributed systems instead of traditional non-distributed-programming, by using overlay networks as an enabler, and combine component-based system with overlay network technology into a service architecture for large-scale distributed system self management instead of traditional focus on individual autonomic properties.

## 5 Emergent topic-Desktop Grids

Actually, the research topics of the Institute in WP4 cover very widely. Except the work that mentioned above, other emergent topics in desktop grids such as trust management, availability evaluation, scheduling strategy and so on are also discussed in detail. Among all these researches, one is focused on the topic of trust-management and sabotage tolerance in grid applications that execute on open environments, like the Internet. The second focus on an experimental study about the availability of nodes in enterprise desktop grids, by using some real traces that were collected in four different infrastructures. And the third research presents another experimental study about scheduling algorithms and techniques to optimize the turnaround time of applications that run in desktop grids. In the overall we can divide these three papers in three different sub-topics.

## 5.1 Trust Management

The research results of the institute in WP4 for the deliverable D.SA.02 related to trust management in desktop grids have been presented in [15], in which two descriptions of the state-of-the-art: an overview about the existing papers in sabotage tolerance and trust management for desktop grids, including replication schemes, encryption-based techniques, sampling and checkpoint-based techniques are presented. After the overview the authors describe two novel techniques: a mechanism for sabotage-detection and a protocol for distributed trust-management, which are better suited to the paradigm of volunteer-based computing commonly deployed in desktop grids. To early detect and finer locate the errors in volunteer computations, they targeted public computing projects, propose the compare replicated checkpoint hashes technique, and complement it with trickle messaging to permit early detection of divergent computations and the occurrence of malicious failures or sabotage attempts. This technique was called by HCV (Hash-checkpoint verification) and apparently it brings some interesting advantages when compared with other existing techniques of the literature. In the second part of the paper, the authors present a summary of the existing work about reputation systems for grids and peer-to-peer systems. In the context of that overview they presented a new idea for the development of a new reputation system, the volunteer invitation-based system (VIS), that is basically based on the notion of explicit invitations among the users and in the maintenance of a dependency tree to support the dynamics of a trust-list. Both overviews presented in this paper can be a good starting point for the researchers that want to contribute to this important topic of trust and security in desktop grids. The techniques that were presented in this paper demonstrate some potential and should be further evaluated by an experimental study.

## 5.2 Availability Evaluation

The research of the institute in WP4 concerning availability evaluation in desktop grids presents an interesting study about the expected availability of computing nodes in enterprise grids [10]. In this research, the authors start by presenting the trace-base method that has been used and the data traces that have been collected and are now in public-domain. By collecting characterization of the traces in four real desktop grids, including metrics like: the number of hosts available over time; the temporal structure of availability and unavailability; the measured rates for task-failure; the existing correlations between the nodes, the authors describe aggregate and per host statistics that reflect the heterogeneity and volatility of desktop grid resources, develop a performance model for desktop grid applications for various task granularities using their characterization and quantify the utility of the desktop grid relative to that of a dedicated cluster for task-parallel applications using a cluster equivalence metric. Different from the traditional CPU availability test methods, the authors put forward a new method used for CPU availability traces, that is, they gather traces by submitting measurement tasks to a desktop grid system that

perceived and executed as real task. The advantage of this method is that the measurement of consume of CPU cycles is a real application. To characterize the availability for task execution, the authors collect data set from two desktop grids, one consists of desktop PC's at the San Diego Super Computer Center (SDSC), the other consists of desktop PC's at the University of Paris South. The observed results show that the total number of host available over time to determine at which times during the week and during each day machines at the most volatile, the successful completion of a task is directly related to the size of availability intervals, the task rate increases with task size almost linearly, the exec availability in desktop grids is not significant correlated from on machine to another, and only weak negative correlation exist between host speed and task failure rate increasing generally with the task size as expected. Considering the temporal of availability directly impacts task execution, the authors also study the exec availability in terms of the CPU availability as perceived by a desktop grid application of the entire system, and give an example of how one can use their characterization of a derived performance model of a desktop grid system, quantifying the negative impact that heterogeneous and volatile resources have on system throughput.

### 5.3 Scheduling

The research results of the institute for the deliverable D.SA.02 related to schedule strategy in desktop grids have mainly been presented in [6], which presents scheduling techniques to optimize the turnaround time of master-worker applications that run in desktop grids. For that purpose, the authors presented a set of techniques, combine shared-checkpoint management with several scheduling methods like FCFS, adaptive timeout, simple replication and short-term availability predictions, aim at minimizing turnaround time by reducing negative effects of resources volatility on the executions of bag-of-tasks applications. Then they present and assess several scheduling policies oriented toward fast turnaround times, and assess them through trace-based simulations over the resources of 32 machines of two classrooms of an academic environment. Together with those scheduling techniques the authors also defend the idea of having a storage service to share the checkpoint images of partial executions among the nodes in order to restart the tasks from previous checkpoints in different nodes, selected by the scheduler, and this way they would be able to get some interesting optimizations in the final turnaround time of the applications. To verify the feasibility and availability of their thought, the authors adapt trace-driven simulations technique in real desktop grid systems to assess the devised scheduling policies such as FCFS-AT, FCFS-TR and FCFS-PRED based on the scheduling methodology by adding the support for shared checkpoints. The results presented in the paper might be of particular interest from the developers of grid middleware, like BOINC, XtremWeb or Entropia. For to every scheduling policy, the shared-checkpoint version almost always yields a better turnaround time than the private version due to the fact that a shared-checkpoint scheduler holds a global view of the system. Thus, the scheduler knows about the whole

state of the system, and consequently can react much more promptly when a failure occurs, by rescheduling tasks accordingly to availabilities.

The second and the third achievements of the institute in WP4 are somehow related since the traces collected in [11] could also be used by the prediction-based techniques and the scheduling heuristics presented in [6]. Some more joint work should be done in these related topics by those authors.

The research [11] investigates the use of buffering to ensure task completion rates, which is essential for soft real-time applications. They develop a model of task completion rate as a function of buffer size, and study it using parameters derived from two enterprise desktop grid data sets. Different from BOINC, XtremWeb, and Entropia, in the proposed model, the scheduler can delete task from the buffer if a task's deadline passes or the buffer overflows with too many tasks, and once a task is scheduled, the scheduler does not have the ability to cancel an executing task. To make their model available, the authors instantiate parameters such as normality, estimating deadline failure rates and so on from two data set collected from two desktop grid system, one is UC Berkeley trace data set and the other is collected from desktops at San Diegao Supercomputer Center. In the last part, the authors present an evaluation via trace-driven simulation, and showing how the proposed model can be used to ensure application task completion rates on enterprise desktop grid systems.

## 6 Comparison of the Joint Work vs. the WP4 Roadmap Version 1

In the roadmap version 1 of WP4 the following three research topics were identified which should deliver architectural components for scalable, adaptive and dependable Grids:

- "Scalable GRID Services",
- "Mechanisms for Adaptive GRID",
- and "Dependability in GRIDs".

In the following we list the research issues within each topic, and map the suggested focus areas to scientific work conducted by the WP4 partners.

### 6.1 Scalable GRID Services

In this task, the following research issues have been identified:

- Scalable Grid Services
- Scalable Storage and Publishing Systems
- Resource Discovery and Searching
- Resource Access and Service Provision

- Scalability vs. Reliability
- Scalability vs. Consistency
- Scalability in Insecure World
- Multicast Services
- Advance Resource Location Mechanisms
- Distributed Catalogue to store Metadata

Based on the issues identified, the following focus areas were recommended and investigated:

- Resource naming based on research conducted in Peer-to-Peer systems; Resource discovery based on research conducted in Peer-to-Peer systems.
  - Lehtonen et al. in [12] presented Boris Object Request InfraStructure (BORIS) – a light-weight P2P-based resource discovery infrastructure, tailored in particular for mobile user applications.
- Fully decentralized models of Grid services based on Peer-to-Peer service invocation; Peer-to-Peer models for the implementation of scalable Grid systems and applications.
  - Glynn et al. in [7] discussed an enhanced structured P2P middleware suitable for implementation of scalable Grid systems and applications.
  - Van Roy et al. in [14] presented a vision self-managing scalable services based on P2P overlay infrastructure deploying software components.
  - Caromel et al. in [3] presented the work towards factorization of component services dealing with non-functional aspects of components, so that those can be deployed independently of the component' main functionality.
- Design and implementation of lightweight Grid services for large scale pervasive Grids; Naming systems for mobile Grids; Search and discovery techniques based on structured and unstructured Peer-to-Peer models.
  - Papadakis et al. in [13] proposed an approach to significantly reduce the overhead of flooding-based search in unstructured networks, which can be useful in highly volatile and/ or heterogeneous Grid environments.

The following recommended focus areas were not investigated as joint work:

- Scalable user interfaces for Grids with information synthesis and automatization of Grid tasks; Reliability and its tolerance to resource variation; Scalable dynamicity of Grid middleware; Efficient distant execution and data diffusion in heterogeneous large scale architectures; Simple distributed security infrastructure in Peer-to-Peer systems; Multi-consistency support on mutable resources in Peer-to-Peer systems; Handling of dynamic information and its influence on scalability and consistency; Enhancement of structured Peer-to-Peer systems to support range queries; Efficiently load balancing of the catalog on heterogeneous servers; Distribution of the data across the servers in order to efficiently answer queries; Introduction of redundancy in order to make the distributed catalog reliable; Study existing or (if necessary) develop new mathematical models to describe the availability of files in Grid and Peer-to-Peer systems; Allow efficient partial file access to enable multi-source data access; Study existing or (if necessary) develop new replica placement algorithms.

In addition the mentioned focus areas, the following joint research has been conducted in the area of scalable GRID services:

- Addressing the issue of scalable resource discovery and searching, Groleau et al. in [8] discussed the application of the semantic Grid techniques for resource discovery in large heterogeneous Grids. Trunfio et al. in [18] reviewed the state-of-the art in peer-to-peer models for resource discovery on Grids, including techniques using structured and unstructured networks, as well as using semantic Grid techniques in P2P-based resource discovery.
- Concerning the issue of scalable storage and publishing systems, Comito et al. in [4] described the PARIS (Peer-to-peer ARchitecture for data Integration Systems) that allows semantic interoperability over distributed data.

## 6.2 Mechanisms for Adaptive GRID

In this task, the following research issues have been identified:

- Modelling and Prediction of Demand/Capacity,
- Adaptive Workflows for System Management,
- Composition of Web Services,
- Adaptive Communication Layers in Grid Systems.

Based on the issues identified, the following focus areas were recommended and investigated:

- Modeling and Prediction of Demand/Capacity

- Andrzejak et al. in [1] presented a prediction study on an institutional Grid, and discussed a useful and computationally inexpensive framework for automated behavior prediction.
- Kondo et al. in [11] presented an analytical model for task success as a function of buffer size, and demonstrated its validity through simulation using data from real desktop Grids.
- Kondo et al. in [10] presented a study of job execution traces in 4 desktop Grids, which is essential for simulation and modeling of such platforms.
- Adaptive Workflows
  - Domingues et al. in [6] presented and evaluated several techniques that improve the task turn-around time of bag-of-task applications, focusing on reducing the impact of resource volatility.
- Adaptive Communication Layers in Grid Systems
  - Cadilhac et al. in [2] presented an analysis of relying techniques for execution of unmodified message-passing applications on Grids, focusing on fault-tolerance and the overhead of these schemes (this contribution also occurs in dependability - Fault-Tolerant MPI).

The following recommended focus areas were not investigated as joint research work:

- Composition of Web Services.

### 6.3 Dependability in GRIDS

In this task, the following research issues have been identified:

- Failure Detection and Diagnosis,
- Checkpointing-and-Recovery,
- Fault-Tolerant MPI,
- Dependability for Data Grids,
- Fault-tolerant Global Computing.

Based on the identified research fields, the following focus areas were recommended and investigated:

- Fault-tolerant MPI
  - Cadilhac et al. in [2] presented an analysis of relying techniques for execution of unmodified message-passing applications on Grids, focusing on fault-tolerance and the overhead of these schemes (this contribution also occurs in adaptability - Adaptive Communication Layers in Grid Systems).

- Sabotage tolerance
  - Sousa et al. in [15] surveyed the topics of sabotage-tolerance and trust management, and presented a novel mechanism for sabotage-detection and a protocol for distributed trust-management.
- Dependability benchmarking
  - Tixeuil and Silva in [17] contributed to the theory of dependability by presenting a fault-injection tool for large distributed systems, and the DBGS – Dependable Benchmark for Grid Systems.
  - Tixeuil et al. in [16] reviewed several state-of-the tools for fault injection and dependability benchmarking in grids, and explained the importance of such tools for dependability assessment of Grid-based applications and Grid middleware.
  - Hoarau et al. in [9] presented a fault-injection service capable of injecting faults at specific points during its execution, and to form specific patterns of the workload in order to check the program code under specific conditions.

The following recommended focus areas were not investigated as joint research work:

- Definition of a failure model for Grid; Definition of dependability attributes for Grid applications; Failure detection; Failure diagnosis; Error propagation; Robustness mechanisms at the middleware level; Configuration management; Checkpointing-recovery; Fault-tolerant scheduling; Task-replication; Combine checkpointing and task-replication; Grid-services replication; Reliable wide-area data movement; Protocols for dependable data Grids; Workflows for dependable Grid; Naturally fault-tolerant and scalable algorithms.

## References

- [1] Artur Andrzejak, Patricio Domingues, and Luis Silva. Classifier-based capacity prediction for desktop grids. In *CoreGRID Integration Workshop, Pisa, Italy*, November 2005.
- [2] Michael Cadilhac, Thomas Herault, and Pierre Lemarinier. Message relaying techniques for computational grids and their relations to fault tolerant message passing for the grid. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [3] Denis Caromel, Christian Delbe, and Alexandre di Costanzo. Peer-to-Peer and fault-tolerance: Towards deployment based technical services. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.

- [4] Carmela Comito, Simon Patarin, and Domenico Talia. A P2P architecture for schema-based data integration in decentralized environments. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [5] Georges Da Costa. P2P resources management for grids. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [6] Patricio Domingues, Artur Andrzejak, and Luis Silva. Scheduling for fast turnaround time on institutional desktop grid. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [7] Kevin Glynn, Raphaël Collet, and Peter Van Roy. Maintaining a structured overlay network in a hostile environment. In *CoreGRID Integration Workshop, Pisa, Italy*, November 2005.
- [8] William Groleau, Vladimir Vlassov, and Konstantin Popov. Towards semantics-based resource discovery for the grid. In *CoreGRID Integration Workshop, Pisa, Italy*, November 2005.
- [9] William Hoarau and Sébastien Tixeuil. Easy fault injection and stress testing with FAIL-FCI. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [10] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, and Henri Casanova. Resource availability in enterprise desktop grids. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [11] Derrick Kondo, Bruno Kindarji, Gilles Fedak, and Franck Cappello. Towards soft real-time applications on enterprise desktop grids. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [12] Sami Lehtonen, Sami Pönkänen, and Mika Pennanen. Service and resource discovery using P2P. In *CoreGRID Integration Workshop, Pisa, Italy*, November 2005.
- [13] Charis Papadakis, Paraskevi Fragopoulou, Elias Athanasopoulos, Marios Dikaiakos, Alexandros Labrinidis, and Evangelos Markatos. A feedback based approach to reduce duplicate message in unstructured Peer-to-Peer systems. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [14] Peter Van Roy, Ali Ghodsi, Jean-Bernard Stefani, Seif Haridi, Thierry Coupaye, Alexander Reinefeld, Ehrhard Winter, and Roland Yap. Self management of large-scale distributed systems by combining structured

overlay networks and components. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.

- [15] Bruno Sousa, Patricio Domingues, and Luis Silva. Sabotage tolerance and trust management in desktop grid computing. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [16] Sébastien Tixeuil, William Hoarau, and Luis Silva. An overview of existing tools for fault-injection and dependability benchmarking in grids. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.
- [17] Sébastien Tixeuil and Luis Moura Silva. Fault-injection and dependability benchmarking for grid computing middleware. In *CoreGRID Integration Workshop, Pisa, Italy*, November 2005.
- [18] Paolo Trunfio, Domenico Talia, Paraskevi Fragopoulou, Charis Papadakis, Matteo Mordacchini, Mika Pennanen, Konstantin Popov, Vladimir Vlassov, and Seif Haridi. Peer-to-Peer models for resource discovery on grids. In *Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture*, Paris, France, January 2006.