BABEŞ-BOLYAI UNIVERSITY OF CLUJ-NAPOCA

FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION

PROTOTYPE FOR A COLLABORATIVE SYSTEM IN BUSINESS

- PHD THESIS SUMMARY -

MIRCEA MOCA

ADVISER: PHD ŞTEFAN IOAN NIŢCHI, PROFESSOR

CLUJ-NAPOCA 2010

Contents

Introduction

The term **collaboration** originates in Latin (*collaboratus*), meaning[1] to *labor together*. Based on the types of entities participating in the process, several collaboration types are identified: human-human, human-computer and computer-computer collaboration. Tanenbaum [44] defines the distributed systems as a collection of independent computers that appears to its users as a **coherent single system**. To meet the coherence and unity of the system, the autonomous components need, one way or another to **collaborate** [44]. The way in which the collaboration is designed for a distributed system depends on its architecture. In this thesis we deal only with the computer-computer collaboration type and our goal is restricted to the distributed systems area.

Grid computing roots are dated about three decades ago, in the '80's. The ground for building systems that harness computing power was, at that time, scientific applications with an increasing demand of computing power. At that time, computing power was already provided by computing centers by means of supercomputers, characterized by expensive, homogeneous hardware. The access difficulty, acquisition and maintenance costs of supercomputers provided certain researchers with sufficient incentive to find an alternative solution to obtain large computing power more easily and cheaply. Afterwards, Foster and Kesselman [32] introduce in 1998 the concept of *Grid computing* or simply *Grid*, which assumed interconnecting globally-dispersed commodity hardware into one system to provide on demand computing power. The authors give the first complete definition of the concept and list the functional and architectural requirements of a true Grid system. Hence, a computational Grid is "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." In a subsequent article [19] Foster et al. refines the definition of the Grid to address social and political issues. Several years later, Foster gives a three-points checklist [17] to characterize a computational Grid and argues that the Grid must be evaluated in terms of the applications and **business value** that it delivers, not its architecture.

Alvaro et al. define the Virtual Organizations (VO) in the context of collaborative systems as a set of users and real organizations that provide resources, such as compute cycles, storage space or on-line services, for users to exploit for a **common goal**. Such goal can be inter-organizational business applications like Grid-based supply chains [4].

Scientists were the first to understand the benefits of the Grid and fully used it for their experiments in realms such as physics or engineering. Initially, Grids were built at a limited scale, aggregating resources at organization level, such as universities. Thus, they existed as isolated systems. Since then, a lot of research and financial effort was put to meet the fundamental of the Grid, namely integrating existent Grids into one global network.

Nowadays, at European level the effort is focused on interconnecting the existent national grids into a

[1] As defined in the Meriam Webster Dictionary.

European infrastructure. This effort is led by DEISA[2], which deploys and operates a persistent, production quality, distributed supercomputing environment with continental scope[25]. More, the EGI-InSPIRE[3] project[4], started on 1 May 2010, co-funded by the European Commission for four years, as a collaborative effort involving more than 50 institutions in over 40 countries. The project aims at establishing a sustainable European Grid Infrastructure.

Describing the aims and scope of the FGCS[5] journal, Peter Sloot[6] emphasize that the Grid is a rapidly developing computing structure that allows components of our information technology infrastructure, computational capabilities, databases, sensors, and people to be shared flexibly as **true collaborative tools**.

Stockinger[7] makes an analysis [42] on Grid technologies, revising its characteristics and goals in the aim to reach the original concept of Grid. Stockinger also identifies the emerging opportunity of Grid to the **business** realm. In this context, the author presents a list of drawbacks for both Grid software and infrastructure. Hence, the software is not yet stable and ready for production use, and security, accounting and billing systems must be in place to make the Grid useful for commercial use [43].

Desktop Grids computing follow a different approach in terms of infrastructure. Unlike traditional Grids which span over one or more trusted administrative domains, Desktop Grids integrate resources from PCs[8] spread over the Internet. This is a different paradigm which allows building a system at global scale much easier, such as BOINC[9] projects. Nevertheless, the quality of the aggregated resource is poor because it is untrusted, highly volatile and heterogeneous. Therefore, the Desktop Grid has its losses, and according to Stockinger [42] they address applications where data or computing results can be lost or easily reproduced.

Some Grid computing systems overlay a peer-to-peer (P2P) structure. With some exceptions, in P2P architectures, participants have relatively equivalent functionality. Because of the permissive communication patterns in P2P systems, P2P computing is considered as a powerful tool to organize Grid and cooperative computing [46]. The fact that Grid and P2P share a common destination makes the research of these two fields combine naturally [29]. In 2003, Foster and Iamnitchi [18] envisioned the merge of these fields to find synergy. Nowadays, a significant effort in this sense is led by the ANL[10] and IBM[11], which have proposed a new Grid research model named OGSA[12] [20], which merged the concept of Grid, P2P and Web Service all together and aims at sharing everything as web services. For example, Grid and P2P are used together in a hierarchical model to build a **collaborative application framework** [49].

The highlighted issues that the Grid and P2P computing is still open and challenging. Further innovation is required to achieve a higher degree of collaboration in these distributed systems. This thesis contributes to enhancing and improving collaboration in Grid and P2P computing systems by employing

[2]Distributed European Infrastructure for Supercomputing Applications.

[3]European Grid Infrastructure.

[4]Contract number: RI-261323.

[5]Future Generation Computer Systems, The International Journal of Grid Computing and eScience.

[6]Peter Sloot is full professor of Computational Science at the University of Amsterdam, editor-in-chief of two Elsevier Science Journals: JoCS (The Journal of Computational Science) and FGCS (Future Generation Computing Systems).

[7]Heinz Stockinger is a scientist and Grid specialist at the Swiss Institute of Bioinformatics. He received his PhD in computer science and business administration from the University of Vienna, Austria. He worked many years for CERN (The European Organization for Nuclear Research) within the EU DataGrid and the EGEE (Enabling Grids for E-sciencE) projects, where his research work focused on Grid data and resource management.

[8]Personal computers.

[9]BOINC: Berkeley Open Infrastructure for Network Computing.

[10]Argonne National Laboratory, one of the U.S. Department of Energy's oldest and largest national laboratories for science and engineering research [23].

[11]The IBM (International Business Machines) company.

[12]Open Grid Service Architecture.

fundamental concepts and methods from economics and business. With such tools, we show that better, more robust and reliable automated collaboration can be achieved, even in distributed systems that spans over multiple administrative domains and conflicting ownerships. To the end, better technology emerges, to the benefit of the business sector.

This summary comprises two chapters, each chapter representing distinct contributions on the topic of collaborative computing systems. In Chapter 1 we improve the resource aggregation process for a distributed computing system with a P2P structure by employing decision models. Such mathematical methods already play an important role in the Economics realm and business [30]. Thus, the aggregation of the resources is performed based on a decisional process. The attributes employed by the decision models are various quality parameters of the computing resource: CPU power, bandwidth, storage capacity etc. We also make a comparative analysis of the main categories of decision models. A software module which such provides automate decision-making functionality can be integrated in a Grid middleware to improve the resource aggregation process.

Business Intelligence is a collection of data warehousing, data mining, analytics[13], reporting and visualization technologies, tools, and practices to collect, integrate, cleanse, and mine enterprise information for decision making [9]. As the size of data sets being collected and analyzed in the industry for Business Intelligence is growing rapidly [3], the traditional data warehouse and other systems that process the data sets become prohibitively expensive. In the last years, systems that implement the MapReduce paradigm [10, 21] emerge as a cost-effective infrastructure alternative for Business Intelligence. This provides with significant aggregated storing and processing capacity the algorithms for datamining, classification, text analysis, collaborative filtering and others.

MapRedue continuously gains ground and more and more researchers find interest in developing systems for supporting Business Intelligence. For instance, Hadoop MapReduce is a software framework for writing applications that rapidly process vast amounts of data in parallel, on large clusters of compute nodes [47]. X-RIME[14] is a cloud-based library for large scale social network analysis, built on top of Hadoop [48]. According to the authors, X-RIME can be integrated with data warehouse systems and datamining tools for clouds, to produce more comprehensive and cost effective social network aware Business Intelligence solutions.

Another technology for distributed computing is DryadLINQ [28, 50], which employs DAG[15]-based execution flows. However, this is shown to have generally weaker performance, compared to Hadoop [12, 13].

However, investing in a cluster - or *company grid* as called by Stockinger, assumes buying new commodity hardware. More, significant costs occur for electricity and personnel to maintain the system. For companies where the system is not sure to be amortized, investing in a cluster would be economically unjustifiable. Thus, at least temporarily, considering the resources supplied by a Desktop Grid infrastructure could be an alternative.

Therefore, in Chapter 2 we introduce an infrastructure for Business Intelligence applications. We present a prototype for MapReduce in a Volunteer Computing environment, built on top of a Desktop Grid infrastructure, using the BitDew[16] middleware. To our knowledge, there has been no such similar approach in the literature, yet. We discuss the key-points of our prototype and present the software requirements, design, implementation and deployment phases. We also give the results of mechanisms and performance

[13] Analytics is the application of computer technology and statistics to solve problems in business and industry [33]

[14] This project is a join effort of Beijing University of Posts and Telecommunications (BUPT) and IBM China Research Lab, supported by IBM Open Collaboration Research program. [26]

[15] DAG: Directed Acyclic Graph

[16] BitDew : Open Source Data Management for Grid, Desktop Grid and Cloud Computing, a project founded by Gilles Fedak and Haiwu He, at INRIA Futurs [24, 15].

evaluation of our system, showing that our proposed solution provides massive fault tolerance, replica management, barrier-less MapReduce, latency-hiding mechanism, dedicated two-level scheduler as well as distributed result checking.

Chapter 1

Decision Models in Resource Aggregation

As service-oriented systems emerge toward a fully decentralized collaborative environment, resource aggregation becomes one of the important features to study. In this chapter we investigate the effectiveness of resource aggregation in a peer-to-peer architecture with autonomous nodes that can either supply or consume services. We consider various setups concerning the initial endowment of the system with resources, the load with service requests, the intrinsic capability of the system for resource discovery and the subjective valuation of peers concerning the delivered services. We show that for high loads of the system with service requests, the performance of the resource supply does not degrade in the long run and for low loads the resource discovery method combined with the partner selection algorithm succeeds to deliver a better performance.

1.1 Introduction

As the grid emerges toward fully distributed P2P networks [18], service oriented architectures need to adapt to the new peer-to-peer networked environment. To make the P2P-based SOA pervasive, the challenge is to let all the nodes in the system to play both roles: consumers and producers of services. Such an ideal system should be able to discover and aggregate the suitable resources to supply a consumer query.

Therefore, a system designer faces several major challenges when building such a system: which is the suitable underlying structure of the P2P network, which mechanism to employ for resource discovery, what model to apply in order to select the proper providers or whether the designed mechanisms can lead to scalable, stable and reliable SOA environments.

On the other hand, agent research contributes with resource allocation mechanisms [7], emphasizing on various issues on interest like agent preferences, production of the social welfare, complexity, negotiation, algorithm and mechanism design etc. But, up to now, very few research concerns whether those conceptual models are effective in fully distributed P2P networked environments.

In this chapter we first[1] investigate the effectiveness of the resource aggregation in unstructured decentralized P2P networks with autonomous nodes. By resource aggregation we understand the process of gathering quantities of the same resource from several providers. We let each node to be either a service consumer or a resource provider, to be equipped with some decision making model and to have different preferences over the decision criteria employed for partner selection. We will draw out some conclusions about how the global performance of the aggregation is affected by the network topology and size, the demand load and the power of the resource discovery mechanism. Further, we investigate whether a

[1] Having in mind the idea of improving the resource aggregation in an unstructured decentralized P2P network by applying decision models, our first goal will be to prove the feasibility of this approach.

consumer-tailored subjective decision making model can bring global gains on the system.

Secondly[2], we investigate several more complex decision models. In [38] we employed only the Onicescu decision model [27], focusing on both the objective and the subjective variants of the model. The Onicescu model fits in the non-parametric class of decision models, the literature [1, 16] recommending also other more complex alternatives. Thus, besides Onicescu, we considered the Global Utility Method [27] from the Multi-Attribute Utility Theory (MAUT) [11] and Promethee [6], covering all spectrum of decision models types. Our goal is to recommend the decision model class that suits best for a given wealth endowment of the P2P network and a given service request load.

This chapter is organized as follows. In Section 1.2 we describe the P2P system model and the parameters and measures used to evaluate the effectiveness of the resource aggregation scheme. Section 1.4 presents our simulations and reports the obtained results and presents a comparative analysis of the mentioned decision models. Section 1.5 summarizes this chapter.

1.2 Background

The P2P System Architecture. We present in this section the system architecture and the aggregation mechanisms employed in our setting.

The discussed system comprises a set of N participants, organized in a unstructured peer-to-peer architecture. Each peer owns a certain quantity of resources and it is linked to a subset of other peers, called neighbors. Consequently, our system is a connected graph. This structure is established a-priori, in the sense that it remains stable during one round of experiments. Thus, before each run, we randomly build the graph structure of the system architecture by selecting the neighbors of each peer.

Each peer p_i owns a quantity q_i from some resource R. The resource R (which might be a service) is defined by a set of issues (properties) $\{is_1, ..., is_k\}$ that characterize the resource. These issues might be the price, the resource quality etc. and can hold numerical values $\{v_{i,1}, v_{i,2}, ..., v_{i,k}\}$, specific for the resource provider p_i. For the sake of simplicity, as the goal of this chapter is to investigate resource aggregation, we endow the system with only one sort of resource R and we vary the issues values.

In our experiments we consider various endowments of the system with resources. Thus, the wealth can be uniformly distributed among the peers or they might be unequally equipped with resources (e.g. some peers might own a big quantity and ask for a higher price in contrast with other peers that can supply only with a small quantity of resource).

Upon this peer-to-peer infrastructure we construct the resource aggregation functionality employing two mechanisms: resource discovery and service composition. During resource discovery, the process starts at a node - called *initiator* - that demands a quantity Q_d of the resource R for T_d units of time. We assume the network is equipped with some resource discovery mechanism that search network in order to discover potential resource providers [45]. The resource discovery mechanism has some intrinsic discovery power in the sense that it is able to investigate a fraction f of the total number of peers. Later in this chapter, we will address the models for resource discovery. The resource discovery process returns a list of potential providers.

Next, the service composition mechanism is applied after resource discovery. During service composition, the initiator selects the proper peers to aggregate resources from, by filtering the list returned by the resource discovery mechanism. If the initiator can not aggregate the entire demanded quantity, the query fails.

[2]After we proved that decision models can be applied to improve the resource aggregation process in an unstructured decentralized P2P system, we continued our research work [37] with a comparative analysis of several decision models from the relevant categories presented in literature [1, 16].

The time T_d related with a resource demand indicates the duration in time units for which the initiator occupies the selected resource during service consumption. When a provider p_i enters a transaction for a resource demand with T_d, during those time units the provider will not be able to commit the resource for another query, thus eliminating the possibility of distributed deadlocks in the system. For simplicity, we assume that each initiator can estimate the time T_d for a resource demand, and if T_d is not enough to supply the initiator's needs, the initiator will launch a new query for the additional required duration.

Models for resource discovery in peer-to-peer architectures are presented in [45]. Among them, we can consider message broadcasting. With message broadcasting, each resource query is broadcasted by the initiator in the network with a time-to-live parameter TTL. The TTL is strongly related with the connection degree of the network. They determine the number of nodes reached by the search - the query horizon. The bigger the TTL, the further the message is delivered in the network, and the query horizon of the resource discovery mechanism increases. The theoretical query horizon can be deduced out of the network size, topology and TTL. The actual horizon is the total number of distinct nodes that actually respond the queries. The theoretical horizon calculation is often irrelevant [41] since it does not take into account graph cycles and variable connection degree at different hops. Hence, we employ in our study the actual horizon, and for simplicity refer with horizon. Being tracked and reported by the system, this value is accurate.

For our experiments, we employed the deterministic simple-flooding broadcasting protocol [36]. Broadcasting is very suitable for our needs because there is a direct relation between the query horizon and the size of the TTL parameter.

In this paragraph we describe the experimentation setup employed in our study. A round of experiments consists of multiple resource demands, each being delivered at individual time units on the time scale. For a resource demand, a peer p_i is randomly selected and it initiates a query for Q_d quantity of resources with T_d. The resource discovery mechanism retrieves a list of potential providers. Next, the initiator applies some decision model in order to select the peers to aggregate resources from. The efficacy of the selection is evaluated and next, the transaction happens in the sense that the selected peers will have the selected quantity of resource unavailable for the next T_d units of time. This resource demand scenario is applied several times and at the end, we report the total efficacy achieved.

During experimentation (Section 1.4) we change various inputs and we report and conclude about how effective the resource aggregation procedure described above is. Next, we describe the decision models used for service composition and the metrics employed to evaluate the efficacy.

1.3 Evaluating the Quality of Resource Discovery

In this section we present the evaluation criteria to assess the effectiveness of the resource aggregation process.

Evaluation should be done individually at the level of each resource demand and globally at the level of all resource demands covered is a experimentation run.

At the end of the resource aggregation process the initiator holds an optimal list with partners as the result of the query injected into the system. As available resources permanently fluctuate in the system in terms of provider and quantity, consecutive demand queries would provide different results. Thus, at the resource demand level, we evaluate the utility perceived by the initiator concerning the result delivered by the system in response to its query. A selected result consists in a number N_p of selected partners, the total prices P_i, $i = \overline{1, N_p}$ paid by each partner and the quantities Q_i delivered for the prices P_i. Initiators are interested in:

- aggregating all the demanded quantity Q_d

- minimizing the payments
- minimizing the risks associated with the transaction delivery. In our case, risks increase with the number of partners per transaction.

Eq. 1.1 describes the individual 'utility' associated with a query.

$$U_d = \frac{1}{N_p} \times \frac{1}{\sum_{i=1}^{N_p} P_i} \times \frac{1}{\sum_{i=1}^{N_p} Q_i} \tag{1.1}$$

The bigger the utility scored by a demand d, the better. Utilities can be aggregated over all demands in a run of experiments to obtain the global utility U_g of a system setup. The global utility U_g characterizes the social welfare concept, presented in [7].

Besides the above-described utility, we also count the number of failures to supply the entire demanded quantity and the total payments. From the consumers' point of view, the objective is to minimize the payments. From the providers' point of view, the objective is to maximize the payments.

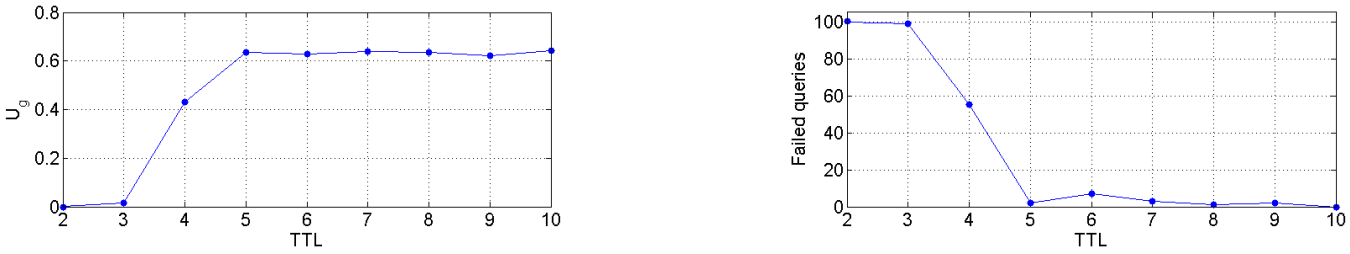
1.4 Experiments and Results

In this section we describe the experiments and comment on the results of our study. The experimentation is performed on a message-based simulator [39] for a P2P network, implemented at the Faculté Polytechnique de Mons Belgium[3] and modified to accommodate the resource aggregation.

We employ the P2P system architecture described in Section 1.2 with the broadcasting protocol for resource discovery. Next, we present the set with the main system parameters that drive our experiments:

- the TTL of the broadcasting mechanism employed for resource discovery,
- the connection degree (D_c), representing the number of neighbors of a node;
- the query horizon (H_d), meaning the number of potential providers that an initiator discovers; this is the practical achieved value for the parameter f - the coverage factor of the resource discovery mechanism
- the number of selected providers (N_p - as in previous section),
- the initial endowment q_i of a node,
- the total number of request messages (N_m) broadcasted for a particular resource demand. This is a cost measure for the resource discovery mechanism;
- the demanded quantity Q_d for a query; might be (i) low, (ii) high or can uniformly vary between the low and the high value. Each fulfilled query will hold the committed resources busy for the next T_d queries, with T_d being set up to a random number from 2 to 10. The demanded quantity is in fact the load factor of the network, as employed in [8].
- and the failure rate (R_f), which is the number of queries that fail within the running of a scenario. A query is considered failed when the initiator ends the resource aggregation procedure without fulfilling all the requested quantity Q_d.

[3]We thank Sebastien Noel from Faculté Polytechnique de Mons Belgium for letting us to use the initial version of the P2P network simulator and for support during the development of our specific version of simulator.



(a) (b)

Figure 1.1: a) Total utility U_g as a function of TTL; b) The failure rate R_f.

We experiment on three different sorts of networks regarding the distribution of q_i. If QN is the total quantity of the resource available in the system, we experiment with networks where:

- $q_i \sim \frac{1}{QN}$ (uniform distribution),
- $q_i \sim Pois(1)$ (Poisson distribution with parameter $\lambda = 1$) - where very few nodes hold large quantities of resources, and
- $q_i \sim Pois(4)$ (Poisson distribution with parameter $\lambda = 4$) distribution, where the majority own the average quantity $\frac{1}{QN}$ and only few nodes own large or small quantities.

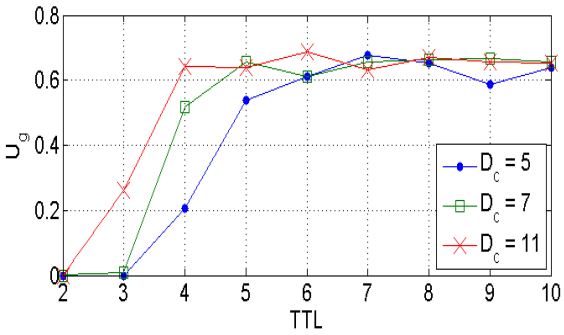
For the rest of our discussion, a scenario (or a run of experiments) is a set of 100 queries initiated by participants randomly chosen from a network of 500 nodes. The results we present below scales proportionally with the network size and the load in queries, by maintaining the same network topology.

We first inspect how TTL influences the global utility since a higher value would lead to a broader horizon. Thus, for fixed values of D_c and Q_d we run scenarios for a range of TTL values. The total utility U_g increases with the TTL, as depicted in figure 1.1a. Figure 1.1b depicts the failure rate R_f decreasing with increasing TTL. We identify three 'stages' for the progress of U_g as we modify TTL. First, for very low values of TTL, U_g is 0, since R_f is 100%. It means the horizon is insufficient for the initiator to discover enough nodes to fulfill Q_d. Starting from a certain TTL, the U_g rises and then becomes stable, while R_f decreases.

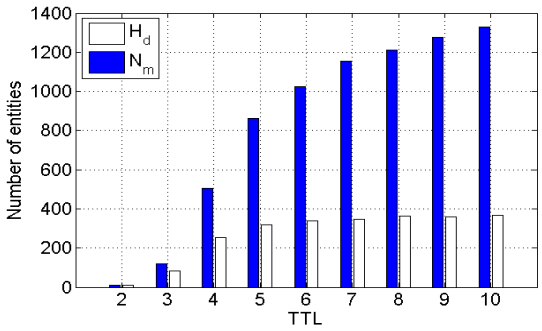
Keeping the same setting, we ran scenarios with different values for Q_d. The pattern followed by the global utility vs. TTL holds as in figure 1.1a. However, for high values of Q_d, U_g begins to increase only with a higher TTL.

The connection degree is another parameter that determines the horizon's ampleness. Thus, for the same scenarios as before: a range of TTL values and fixed Q_d, we inspect U_g while modifying the connection degree D_c. Figure 1.2a show the total utility curve U_g for three distinct values of D_c; Q_d being fixed. We observe that a greater D_c improves U_g, in the sense that stages of increasing and stabilizing emerge sooner, thus for lower values of TTL.

A more detailed analysis of U_g from the communication costs' perspective, conveys us to the assertion that both D_c and TTL are worth increasing up to a certain level. That is, after U_g converged to its upper limit for the given network setup, increasing D_c and TTL lead only to higher costs, without improving the total utility. Figure 1.2b depicts the horizon H_d as it flattens with a certain TTL, although the number of messages continue to increase with TTL. We also note that this behavior holds in scenarios with different values of Q_d and D_c. More, if we simultaneously look at how U_g and H_d modify, we observe that they converge for the same values of TTL and D_c. Consequently, when designing a real system, D_c and TTL should be adjusted to values that maximize U_g and still avoid useless retransmission.



(a) Total utility.



(b) The horizon and number of messages.

Figure 1.2: Gains and costs as a function of TTL, for various connection degrees

In our thesis we present also other experiments for different types of networks, related to the distributions of q_i and also other key parameters. We also compare the two versions of the Onicescu's model: the subjective and objective versions, observing that the aggregated resources can be adapted in order to match initiator's requirements (according to the values specified for the attributes that characterize the resource). This is possible biecaus the subjective version of the model allows the initiator to specify a ranking of importance among the criteria (attributes) when judging the alternatives.

Comparative Analysis of Several Decision Models. In this section we report the results of a comparative analysis which targets several decision models. First, we inspect how the different decision models perform in terms of the global utility as a function of TTL (figure 1.3), as we expect that a higher TTL value would lead to a broader horizon and therefore, a better satisfaction. Thus, for a uniformly distributed load Q_d of the network, we run scenarios for each decision model and a range of TTL values.

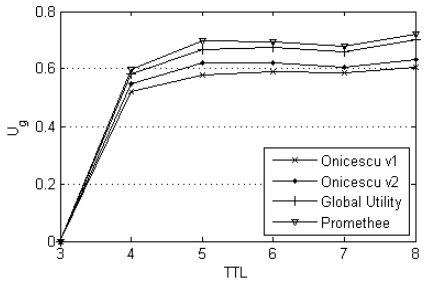
We notice that the total utility U_g increases with the TTL, as depicted in figure 1.3. We also notice that decision aids show similar performances without regard to the distribution of q_i, except the MAUT-based decision aid, which performs differently. Hence, this method performs better when the value of q_i has a Pois(1) distribution, nearly equaling the performance of Promethee aid. For the rest of the distribution types of q_i Promethee yields best results in terms of global utility. We also notice that both Promethee and the MAUT-based parametric method yield better results than Onicescu. This means that is worth to investigate more complex decision models when designing the resource aggregation inside a P2P architecture.

Next, within the same scenario we highlight the variation of two parameters: the query horizon H_d and number of selected participants N_p (figure 1.4). In figure 1.4 we considered the $q_i \approx Pois(4)$ distribution of wealth, corresponding with the situation in figure 1.3c. Query horizon and the number of selected partners is almost the same for the rest of decision models. Thus, it means that Promethee succeeds to improve the global user satisfaction not by minimizing the payments and increasing the monetary benefits but, from reducing the risks associated with the aggregation transactions.

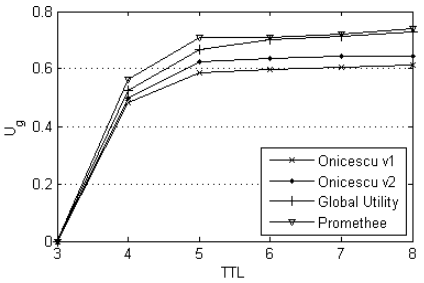
Out of figure 1.4, we extract an interesting conclusion: the Promethee method which supplies with the best user satisfaction achieves it with fewer selected participants in the aggregation (figure 1.4b), which leaves a larger available horizon for the forthcomming queries (figure 1.4a)

Further, we inspect the the way in which the network load influences the global utility U_g. Thus, for a range of TTL values we run experiments with high as well as low values of Q_d. Figure 1.5 presents the results.

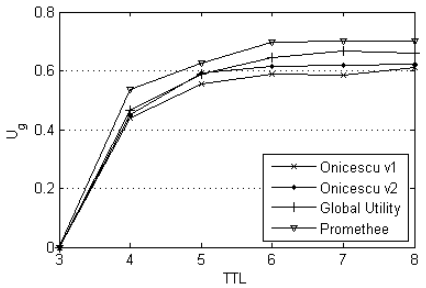
We notice that even for high loads of the P2P network (figure 1.5a), still the Promethee method pro-



(a) $q_i \sim \frac{1}{QN}$.

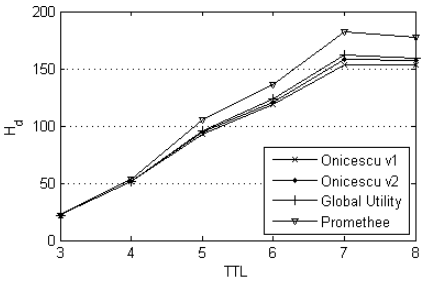


(b) $q_i \sim Pois(1)$.

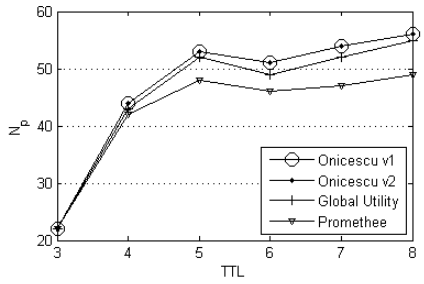


(c) $q_i \sim Pois(4)$.

Figure 1.3: Total utility curves for different distributions of q_i value.



(a) Query horizon.



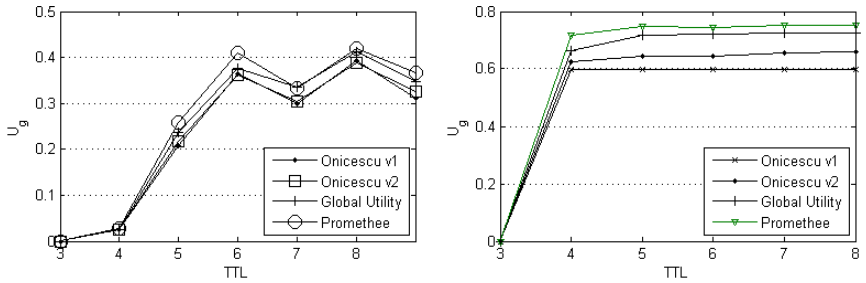
(b) Selected partners.

Figure 1.4: Horizon and selected partners variation.

duces the best results. The low loads of the network (figure 1.5b) shows how weak is a simpler decision model like Onicescu.

In Section 1.4 we emphasized that a subjective decision model has the power to lead to better user satisfaction in heterogeneous environments, characterized by medium to high network loads and un-evenly distributed wealth. This conclusion is further supported by the evidences extracted out of our experimentation, which put the Promethee method in front of the rest. Promethee is strongly characterized by subjectivity, in thesis. In Promethee, each decision maker has its own (subjective) preference relation among the criteria, similar with the MAUT. We also noticed that the parametric MAUT-based model scored better results than Onicescu.

Another conclusion is the fact that Promethee overpasses MAUT indicates that a ranking extracted from local pairwise comparisons performs better than a ranking computed from a global evaluation.



(a) Q_d value is high. (b) Q_d value is low.

Figure 1.5: Global utility for high and low loads of network queries

1.5 Summary

In this chapter we investigated the effectiveness of resource aggregation over a P2P networked infrastructure. Resource aggregation is viewed as the process of collecting quantities from the same resource from various providers.

We concluded that unstructured P2P networks equipped with resource discovery mechanisms with a given horizon are able to properly fulfill the consumers' queries. By experimentation, one can find the proper parameters for the network connectivity and for the resource discovery mechanism. Increasing the connectivity or the power of the resource discovery mechanism more than an optimum will not lead to more satisfaction among consumers. We investigated two variants of the decision making algorithm that put equal emphasis on the decision criteria or consider the decision maker's preference among them. We noticed that when the wealth is heterogeneously distributed on the environment, the variant of the decision making algorithm that differences between the criteria can bring more global welfare.

As a future work we intend to evaluate other decision making algorithms, including parametric ones. Besides aggregation we intend to consider the more sophisticated bundling of resources, where a bundle can comprise quantities from several sorts. Also, reputation of nodes can help as an additional decision criterion during both resource discovery and service composition. The resource discovery costs can be reduced by properly selecting the nodes. Another issue to consider is the budget limitation of the initiators, thus, the demand might be bounded.

Chapter 2

Prototype for MapReduce in Desktop Grid Systems

2.1 Introduction

Desktop Grids have been very successful in the area of High Throughput Computing [35]. They initially supported only Bag-of-Tasks applications, with low disk storage or communication bandwidth requirements and without dependencies between tasks. A broad range of problems of simulating in general, building large indexes or data mining are characterized by significant amounts of input and intermediate data and frequent data reuse. These types of problems can be solved with the MapReduce[1] algorithm, which is discussed in detail in the PhD thesis. Thus, running a MapReduce environment on Desktop Grids would provide a solution framework for the mentioned types of problems.

Given the state of the art in actual Desktop Grids middleware, enabling MapReduce for Desktop Grids is a challenging research field. While traditional Desktop Grids such as BOINC [2], XtremWeb [14] or Condor [35] are tailored for Bag-of-Tasks applications which require few I/O operations, the MapReduce applications handle significant amounts of input and intermediate data, and thus require a different approach.

After the Map[2] phase, the MapReduce model requires collective file operation to handle the intermediate results. Insuring collective communications in Desktop Grids is challenging due to the volatility of hosts and the extremely varying network conditions.

Another challenge is that some components of the Desktop Grid have to be decentralized. For instance, a key security component is the results certification [40] which is needed to insure the correctness of the results of a computation. Because intermediate results might be too large to be sent back to the server, results certification mechanism cannot be centralized as it is currently implemented in existing Desktop Grid systems.

The dependencies between the Reduce and Map tasks, combined with hosts' volatility and laggers can slowdown dramatically the execution of MapReduce applications. Thus, we identify a need to develop a

[1] MapReduce is a programming model and an associated implementation for processing and generating large data sets [10]

[2] In the following of this chapter we will use the terms map and reduce to express different notions. Thus, we use the following types of writing:

Type of writing	Significance
Map, Reduce	functions
Map, Reduce	tasks, algorithm phases
map, reduce	the execution of a Map / Reduce task

performance optimized solution, which answers to the raised challenges.

In this chapter we present a complete runtime environment to execute MapReduce applications on Desktop Grids. At our knowledge there exists no such environment which has been specifically designed for Desktop Grid. Although existing MapReduce middleware such as Hadoop [22] have fault-tolerant capabilities, it is not suitable to the Desktop Grid context [34]. The system as it is, represents the result of the research work performed by the MapReduce working group[3], of which we are part.

Our prototype is based on the BitDew [15] middleware, developed by INRIA, which is a programmable environment for automatic and transparent data management on computational Desktop Grids.

In this chapter we present the design of our system and a set of features which makes our approach suitable for large scale and loosely connected Internet Desktop Grid. These are fault tolerance, replica management, barriers-free execution, latency-hiding optimization as well as distributed result verification. We also show that our system is scalable, achieving a linear speedup for the WordCount application. Several scenarios involving laggers host and numerous host crashes demonstrate that the prototype is able to cope with experimental context similar to real-world Internet.

In the continuation of this chapter we report on the performance evaluation of our prototype, then we conclude in Section 2.3.

2.2 Performance Evaluation

In this section we present the performance evaluation of the MapReduce runtime environment. To measure precisely performances of the execution runtime, we conduct experiments within an environment where experimental conditions are reproducible. We used the *Grid Explorer (GdX)* cluster which is part of the Grid5000 infrastructure [5]. GdX is composed of 356 IBM eServer nodes featuring AMD Opteron CPU running at 2.4Ghz with 2GB RAM interconnected by Gigabit ethernet nertwork. When running the experiments, we have found that most of the nodes had between 5 to 10GB of disk space available. To emulate a Desktop Grid on GdX, we inject faults by killing worker processes and simulate heterogeneity by launching concurrent processes.

2.2.1 Collective Communication

DataCollection and DataChunk are two new features specifically added to BitDew, which have not been benchmarked before. The first experiment aims at determining the optimal chunk size when sending/receiving large file. We run a Ping Pong benchmark using a 2.7GB large file and make the chunk size to vary between 5MB to 2.7GB. In consequence, the number of chunks varies from 540 to 1. The file transfer protocol used to send and receive data is the FTP protocol.

2.2.2 MapReduce evaluation

We evaluate now the performance of our implementation of MapReduce. The benchmark used is the WordCount application, which is a representative example of MapReduce application, adapted from the Hadoop distribution. WordCount counts the number of occurrences of each word in a large collection of documents. The file transfer protocol used to send and receive data is the HTTP protocol.

[3]Bing Tang (Wuhan University of Technology, China), **Mircea Moca** (Babeş-Bolyai University of Cluj-Napoca, România), Stéphane Chevalier (Ecole Normale Superieure de Lyon, France), Haiwu He and Gilles Fedak (INRIA, University of Lyon, France).

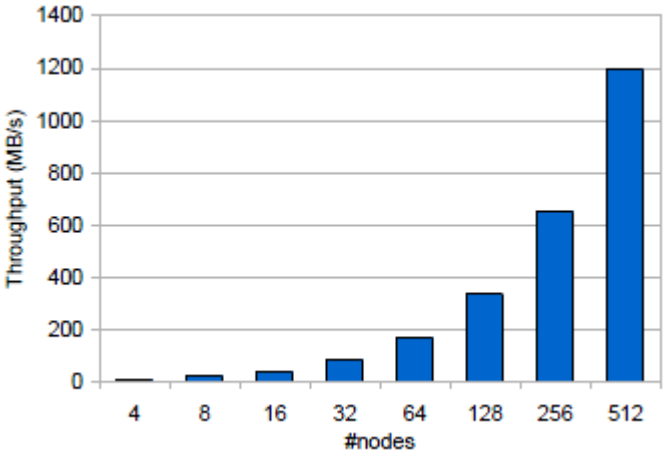


Figure 2.1: Scalability evaluation on the WordCount application: the *y* axis presents the throughput in MB/s and the *x* axis the number of nodes varying from 1 to 512.

#Mappers	4	8	16	32	32	32	32
#Reducers	1	1	1	1	4	8	16
Map (sec.)	892	450	221	121	123	121	125
Reduce (sec.)	4.5	4.5	4.3	4.4	1	0.5	0
Makespan (sec.)	908	473	246	142	146	144	150

Table 2.1: Evolution of the performance according to the number of Mappers and Reducers.

The first experiment evaluates the scalability of our implementation when the number of nodes increases. Each node has a different 5GB file to process, splitted into 50 local chunks. Thus, when the number of nodes doubles, the size of the whole document counted doubles too. For 512 nodes[4], the benchmark processes 2.5TB of data and executes 50000 Map and Reduce tasks. Figure 2.1 presents the throughput of the WordCount benchmark in MB/s versus the number of worker nodes. This result shows the scalability of our approach and illustrates the potential of using Desktop Grid resources to process a vast amount of data.

The next experiment aims at evaluating the impact of a varying number of mappers and reducers. Table 2.1 presents the time spent in Map function, the time spent in Reduce function and the total makespan of WordCount for a number of mappers varying from 4 to 32 and a number of reducers varying from 1 to 16. As expected, the Map and Reduce time decreases when the number of mappers and reducers increases. The difference between the makespan and the Map plus Reduce time is explained by the communication and time elapsed in waiting loops. Although the Reduce time seems very low compared to the Map time, this is typical of MapReduce application. A survey [31] of scientific MapReduce applications at the research Yahoo cluster showed that more than 93% of the codes where Map-only or Map-mostly applications.

2.2.3 Desktop Grid scenario

In this section, we emulate a Desktop Grid on the GdX cluster by confronting our prototype to scenarios involving host crashes, laggers hosts and slow network connection.

The first scenario aims at testing if our system is fault tolerant, that is, if a fraction of our system fails, the remaining participants are able to terminate the MapReduce application. In order to demonstrate this

[4]GdX has 356 double core nodes, so to measure the performance on 512 nodes we run two workers per node on 256 nodes.

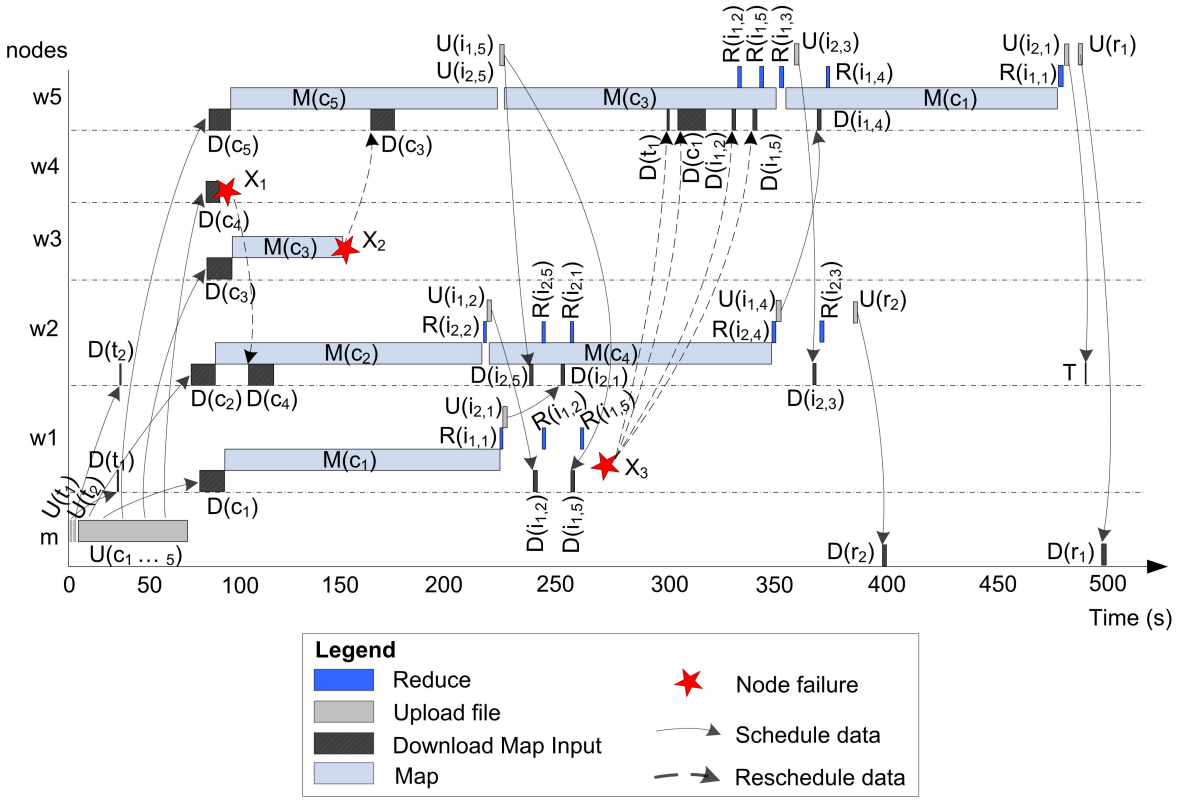


Figure 2.2: MapReduce running with fault-tolerance scenarios.

capacity, we propose a scenario where workers crash at different times: the first fault (F_1) is a worker node crash while downloading a *Map* input file, the second fault (F_2) occurs during the Map phase and the third crash (F_3) happens after the worker has performed the Map and Reduce tasks. We execute the scenario and emulate worker crash by killing the worker process. In figure 2.2 we report the events as they were measured during the execution of the scenario in a Gantt chart. We denote $w_1 - w_5$ the workers and m the master node. The execution of our experiment begins with the master node uploading and scheduling two Reduce token files: U_{t_1} and U_{t_2}. Worker w_1 receives t_1 and worker w_2 receives t_2. Then, the master node uploads and schedules the *Map* input files (chunks) ($U_{C_{1...5}}$). Each worker downloads one such chunk, denoted with $D(C_1)\ldots D(C_5)$. Node w_4 fails (F_1) while downloading *Map* input chunk D_{C_4}. As the BitDew scheduler periodically checks whether participants are still present, after a short moment following the failure, node w_4 is considered to be failed. This conveys to the rescheduling of C_4 to node w_2. Node w_3 fails (F_2) while performing the Map task $M(C_3)$. Then, chunk C_3 is rescheduled to node w_5. At F_3, node w_1 fails after having already performed the Map task $M(C_1)$ and several Reduce tasks: $R_{F_{1,1}}$, $R_{F_{1,2}}$ and $R_{F_{1,5}}$. The notation $R_{F_{p,k}}$ refers to the Reduce task which takes as input the intermediate result F_i, k, obtained from chunk C_k. F_3 causes the BitDew scheduler to reschedule the token file t_1, the chunk C_1 and the intermediates $F_{1,2}$ and $F_{1,5}$ to node w_5.

After finishing the Map task $M(C_1)$, worker w_5 executes the Reduce task $R(F_{1,1})$. Then, it uploads and schedules the intermediate result $F_{1,2}$ to node w_2. Node w_2 downloads the respective intermediate result $D(F_{2,3})$, but without performing any further Reduce task for this input (symbolized with G). This is because every reducer keeps a list with already performed Reduce tasks to prevent our system from processing multiple copies of the same intermediate result.

In our thesis we present also other experiments such as scenarios where we confront our system against the lagger's effect. We inspect how does the presence of laggers influence the makespan of our benchmark

application. In these experiments we consider various weights of laggers in the system. To avoid the effect of laggers we propose a mechanism which verifies the tasks for which results are not reported by workers and reschedules the respective tasks.

2.3 Summary

In this chapter we presented the key issues of the software development process for introducing an implementation of MapReduce for Desktop Grid employing the BitDew middleware. However, in this phase of our research we aimed to prove the feasibility of the MapReduce algorithm in a Volunteer Computing environment, on a Desktop Grid infrastructure. In our discussion we insisted on the software analysis and software design phases giving details about our system.

The features that endow our prototype are massive fault tolerance, replica management, barrier-less MapReduce, latency-hiding mechanism, dedicated 2-level scheduler as well as distributed result checking.

We also performed the evaluation of our prototype, regarding both mechanism's correctness and performance.

The scalability test shows that we achieve linear speedup on the classical WordCount benchmark application. Several scenarios involving lagger hosts and numerous host crashes demonstrate that our approach is suitable for large scale and loosely connected Internet Desktop Grid.

We found that, to be really efficient, input data have to be highly shared between hosts so that P2P protocol can be used to distribute the data or frequently used by tasks in order to increase the computation/communication ratio. Our future work will focus on improving the scheduling heuristics, providing QoS and improving distributed result checking.

In conclusion, we showed that although MapReduce is significantly more complex than traditional Bag-of-Tasks application, it is possible to build a runtime environment efficient and secure to enable data-intensive application on Desktop Grid.

Conclusion

In this summary the discussion was organized in two chapters. In Chapter 1 we proposed the improvement for the resource aggregation phase of the resource management process by applying decision models. In Chapter 2 we proposed a cost-effective infrastructure for Business Intelligence applications. The infrastructure is a MapReduce prototype for a Desktop Grid substrate and a Volunteer Computing environment.

We started by studying the feasibility of decision models for the resource aggregation phase by implementing two versions of Onicescu's algorithm. We found that a decision making algorithm that reflects closer the preference of the users between various decision criteria is most suitable for heterogeneous environments and can take advantage from the heterogeneity.

Afterwards we extended the research by implementing other decision models, including parametric decision models. We provided a compared analysis of different types of decision models and found that a decision model based on pairwise comparisons performs better than models that are based on rankings from early phases. This is because the latter ones loose information which is needed to differentiate between alternatives based on their attribute's values. More, we showed that employing decision models for the resource aggregation phase allows the user to characterize the requested resource, and the aggregated resource will fit better his need.

In Chapter 2 we introduced a prototype for MapReduce for the Volunteer Computing environment on Desktop Grid infrastructure employing the BitDew middleware. The system resulted from the aggregated efforts of an international research team, (introduced in our thesis) from which we are part.

We presented an evaluation of the mechanisms and performance of our prototype. We showed that our system is endowed with the following features: massive fault tolerance, replica management, barrier-less MapReduce, latency-hiding mechanism, dedicated 2-level scheduler as well as distributed result checking.

The scalability test showed that we achieve linear speedup on the classical WordCount benchmark application. Several scenarios involving lagger hosts and numerous host crashes demonstrated that our approach is suitable for large scale and loosely connected Internet Desktop Grid infrastructure.

We found that, to be really efficient, input data must be highly shared between hosts so that P2P protocol can be used to distribute the data or frequently used by tasks in order to increase the computation/communication ratio. Our future work will focus on improving the scheduling heuristics, providing QoS and improving distributed result checking.

In conclusion, we showed that although MapReduce is significantly more complex than traditional Bag-of-Tasks application, it is possible to build a runtime environment efficient and secure to enable data-intensive application on Desktop Grid.

We also presented the key issues of the software development process for our MapReduce implementation for Desktop Grid. However, in this phase of our research we aimed to prove the feasibility of the MapReduce algorithm in a Volunteer Computing environment, on a Desktop Grid infrastructure. In our discussion we put the accent on the software analysis and software design phases giving details of our solution.

Results

The results presented in this thesis were disseminated through the following articles:

Articles published in journals with national scope:

• Mircea Moca, Gheorghe Cosmin Silaghi, *A functional Sketch for Resources Management In Collaborative Systems for Business*, Analele Universităţii din Oradea - Ştiinţe Economice, pp 1447-1453, ISSN 1582-5450, 2008.

• Mircea Moca, Gheorghe Cosmin Silaghi, Tehnologii pentru gestiunea eficientă a resurselor de calcul în medii colaborative pentru afaceri, Studii şi Cercetări Economice, Alma Mater, pp 371-379, ISBN: 978-606-504-035-9, Cluj-Napoca, 2008.

Articles published in proceedings of international conferences held in România:

• Mircea Moca, Gheorghe Cosmin Silaghi, *Analysis On Collaborative Aspects in P2P Architectures*, Annals of the "Tiberiu Popoviciu" Seminar, vol. 6b, International Workshop in Collaborative Systems and Information Society, pp 102-113, 2008.

Articles published in proceedings of international foreign conferences:

• Bing Tang, Mircea Moca, Stéphane Chevalier, Haiwu He, Gilles Fedak, *Towards MapReduce for Desktop Grid Computing*, Fifth International Conference On P2P, Parallel, Grid, Cloud And Internet Computing, 2010, IEEE Computer Society.

• Mircea Moca, Gheorghe Cosmin Silaghi, *Decision Models for Resource Aggregation in Peer-to-Peer Architectures*, Proceedings of the CoreGrid-ERCIM-Working-Group on Grids, P2P and Services Computing held in Conjunction with EuroPar 2009, Delft, Netherlands, pp 105-117, LNCS, Springer USA, 2010, ISBN: 978-1-4419-6793-0.

• Mircea Moca, Gheorghe Cosmin Silaghi, *Resource Aggregation Effectiveness in Peer-to-Peer Architectures*, Proceedings of the 4th International Conference on Grid and Pervasive Computing, Advances In Grid And Pervasive Computing, pp 388-399, Geneva, Switzerland, 2009, LNCS, Springer-Verlag Berlin, ISSN: 0302-9743, ISBN: 978-3-642-01670-7.

• Mircea Moca, *Resource Management for a Peer-to-Peer Service Oriented Computing System*, Proceedings of the 10th European Agent Systems Summer School, pp 31-39, Lisbon, Portugal, 2008.

Technical reports:

• Mircea Moca, Gheorghe Cosmin Silaghi, Gilles Fedak, *Characterizing errors in MapReduce for Desktop Grids*, INRIA Technical Report, Lyon, France, 2010.

Bibliography

[1] M. Abdellaoui and D. Hey, J. *Advances in Decision Making Under Risk and Uncertainty*. Springer, 2008.

[2] P. Anderson, D. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the 5th IEEE/ACM International GRID Workshop*, Pittsburgh, USA, 2004.

[3] T. Ashish, S. Joydeep Sen, J. Namit, S. Zheng, C. Prasad, A. Suresh, L. Hao, P. Wyckoff, and M. Raghotham. Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2):1626–1629, 2009.

[4] L Blasi, AE Arenas, B Aziz, P Mori, U Rovati, B Crispo, F Martinelli, and P Massonet. A secure environment for grid-based supply chains. 2008.

[5] R. Bolze and all. Grid5000: A Large Scale Highly Reconfigurable Experimental Grid Testbed. *International Journal on High Peerformance Computing and Applications*, 2006.

[6] P. Brans, J., M. Mareschal, and P. Vincke. How to select and how to rank projects: the promethee method. In *European Journal of Operational Research*, volume 2, pages 228–238, 1986.

[7] Y. Chevaleyre, E. Dunne, P., U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, A. Rodríguez-Aguilar, J., and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[8] L. Chunlin and L. Layuan. Multi economic agent interaction for optimizing the aggregate utility of grid users in computational grid. *Applied Intelligence*, 25(2):147–158, 2006.

[9] U. Dayal, M. Castellanos, A. Simitsis, and K. Wilkinson. Data integration flows for business intelligence. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 1–11, New York, NY, USA, 2009. ACM.

[10] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, pages 137–149, USA, 2004. USENIX Association.

[11] J. S. Dyer, P. C. Fishburn, R. E. Steuer, J. Wallenius, and S. Zionts. *Multiple Criteria Decision Making, Multiattribute Utility Theory: The Next Ten Years*, volume 38. INFORMS, 1992.

[12] J. Ekanayake, S. Pallickara, and G. Fox. Mapreduce for data intensive scientific analyses. In *ESCIENCE '08: Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pages 277–284, Washington, DC, USA, 2008. IEEE Computer Society.

[13] Jaliya Ekanayake, Thilina Gunarathne, Judy Qiu, Geoffrey C. Fox, Scott Beason, Jong Youl Choi, Yang Ruan, Seung-Hee Bae, and Hui Li. Draft report: Applicability of dryadlinq to scientific applications. 10/16/2009 2009.

[14] G. Fedak, C. Germain, V. Néri, and F. Cappello. XtremWeb: A Generic Global Computing Platform. In *Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid CCGRID'2001, Special Session Global Computing on Personal Devices*, pages 582–587, Brisbane, Australia, May 2001. IEEE/ACM, IEEE Press.

[15] G. Fedak, H. He, and F. Cappello. BitDew: A Data Management and Distribution Service with Multi-Protocol and Reliable File Transfer. *Journal of Network and Computer Applications*, 32(5):961–975, September 2009.

[16] J. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, 2005.

[17] I. Foster. What is the grid? - a three point checklist. *GRIDtoday*, 1, July 2002.

[18] I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Peer-to-Peer Systems II, Second Intl. Workshop, IPTPS 2003*, volume 2735 of *LNCS*, pages 118–128. Springer, 2003.

[19] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Jounral of Supercomputer Applications*, 15, 2001.

[20] I. Foster and C. etc. Kesselman. The physiology of the grid: An open grid services architecture for distributed systems integration. 2002.

[21] Greenplum. Greenplum mapreduce: A unified engine for rdbms and mapreduce, 2010.

[22] http://hadoop.apache.org/core/.

[23] http://www.anl.gov.

[24] http://www.bitdew.net/.

[25] http://www.deisa.eu/.

[26] http://xrime.sourceforge.net/.

[27] L. Ilies, M. Mortan, D. Lungescu, I. Lazar, M. Popa, and V. Vereş. *Handbook of Management (in Romanian)*. Risoprint, 2006.

[28] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 59–72, New York, NY, USA, 2007. ACM.

[29] S. Jiulong, C. Huaping, S. Guangzhong, and C. Xin. Vast: A service based resource integration system for grid society. In Jiannong Cao, Laurence Yang, Minyi Guo, and Francis Lau, editors, *Parallel and Distributed Processing and Applications*, volume 3358 of *LNCS*, pages 489–498. Springer Berlin / Heidelberg, 2005.

[30] S. Karlin. *Mathematical methods and theory in games, programming, and economics*, volume III of *Addison-Wesley series in statistics*. Pergamon Press, London, 1959.

[31] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan. An analysis of traces from a production mapreduce cluster. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, 2010.

[32] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[33] R. Kohavi, J. Rothleder, N., and E. Simoudis. Emerging trends in business analytics. *Commun. ACM*, 45:45–48, August 2002.

[34] H. Lin, J. Archuleta, X. Ma, and W. Feng. Moon: Mapreduce on opportunistic environments. In *HPDC*, Chicago, 2010.

[35] J. Litzkow, M., M. Livny, and W. Mutka, M. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS)*, pages 104–111, Washington, DC, 1988. IEEE Computer Society.

[36] I. Mkwawa and D. Kouvatsos. Broadcasting methods in mobile ad hoc networks: An overview. In *Technical Proc. of the Third Intl. Working Conf. HET-NETs 2005*, pages T9/1–14. Networks UK, 2005.

[37] M. Moca and G. Silaghi. Decision models for resource aggregation in peer-to-peer architecturess. In *Proceedings of the CoreGrid-ERCIM-Working-Group on Grids, P2P and Services Computing held in Conjunction with EuroPar 2009*, pages 105–117. SPRINGER, 233 SPRING STREET, NEW YORK, NY 10013, UNITED STATES, 2009.

[38] M. Moca and G. Silaghi. Resource aggregation effectiveness in peer-to-peer architectures. In *Proceedings of the 4th International Conference on Grid and Pervasive Computing, ADVANCES IN GRID AND PERVASIVE COMPUTING*, pages 388–399. SPRINGER-VERLAG BERLIN, 2009.

[39] S. Noel, P. Manneback, and C. Silaghi, G. Response deadline evaluation in point-to-point negotiation on grids. In *Grid Economics and Business Models workshop GECON 2009*, volume 5745, pages 15–27, Delft, Netherlands, 2009. Lecture Notes in Computer Science.

[40] F. G. Sarmenta, L. Sabotage-Tolerance Mechanisms for Volunteer Computing Systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.

[41] Clip2 Distributed Search Services. *Gnutella Dynamic Query Protocol v0.4*. http://www.stanford.edu/class/cs244b/gnutella_protocol_0.4.pdf, 2003.

[42] H. Stockinger. Grid computing: A critical discussion on business applicability. *J. Supercomput.*, 7(6):3–17, 2006.

[43] H. Stockinger. Defining the grid: a snapshot on the current view. *J. Supercomput.*, 42(1):3–17, October 2007.

[44] S. Tanenbaum, A. and V. Steen, M. *Distributed Systems Principles and Paradigms*. Pearson Education Inc., Upper Saddle River, New Jersey, USA, 2007.

[45] P. Trunfio, C. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-peer resource discovery in grids: Models and systems. *Future Generation Computer Systems*, 23(7):864–878, 2007.

[46] T. Wanqing and J Weijia. End host multicast for peer-to-peer systems. In *Grid and Cooperative Computing 2004, LNCS 3251*. Springer-Verlag Berlin Heidelberg, 2004.

[47] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media Inc., Sebastopol, CA 95472, first edition, 2009.

[48] W. Xue, J. Shi, and B. Yang. X-rime: Cloud-based large scale social network analysis. In *Proceedings of the 2010 IEEE International Conference on Services Computing*, SCC '10, pages 506–513, Washington, DC, USA, 2010. IEEE Computer Society.

[49] X. Yingjie, Z. Mingzhe, and Z. Nengneng. A collaborative application framework based on the p2p grid model. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining*, WKDD '10, pages 63–66, Washington, DC, USA, 2010. IEEE Computer Society.

[50] Y. Yuan, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. Kumar, and J. Currey, G. Dryadlinq: A system for general-purpose distributed data-parallel computing using a high-level language, 2008.