Toward Agent-based Cooperative Resource Management in a Telecommunication Operator Grid Platform

A. Lenica, F. Ogel
MAPS/AMS Group
France Telecom - R&D Division
92794 Issy les Moulineaux, France
firstname.name@francetelecom.com

F. Peschanski, J.P. Briot
Laboratoire d'Informatique Paris VI
Université Pierre et Marie Curie
75252 Paris Cedex 05, France
firstname.name@lip6.fr

Abstract

Like most telecommunication companies and service providers, France Telecom deals with highly heterogeneous applications, ranging from batch oriented to rich interactive multimedia services. Deploying those applications on an utility/grid infrastructure is a promising way to cut operational costs while improving performance (load limitations, QoS, etc.). However, most grid middleware technologies do not deal with interactive (i.e. non-batch) applications. Thus, we propose a grid service platform that dynamically provisions resources for both interactive and batch applications to meet their QoS constraints while ensuring good resource mutualization. Moreover, we believe that relying on an agent-based approach for resource management allows a more flexible, robust and scalable solution.

1 Introduction

As many service providers and telecommunication companies, France Telecom deals with a large number of heterogeneous applications ranging from batch-oriented (e.g. magnetic field and radio simulations, fraud and churn analysis through datamining) to interactive services (e.g. realtime billing, VoIP, videoconferences). For now, most of these applications are deployed on top of isolated clusters, statically over-provisionned to handle peak loads. As a consequence, resource usage is low (most often between 20% and 30%) and administration as well as maintenance costs are high. Adopting a grid-like infrastructure to factor resources represents an efficient way to optimize both resource usage and operational costs. Whereas most grid computing efforts have focused on batch-oriented, massive computational problems, a growing community is investigating the potential of grid infrastructures for more interactive, business-oriented or multimedia applications as well. We focus on the correct and efficient management of resources. Interactive applications, for instance, exhibit dynamic and stringent QoS requirements. Hence, resource management must be very flexible and dynamic. The allocation policies must adapt to the dynamically changing environment in which the applications evolve. One also has to ensure the continuity of service, raising dependability and availability issues. Last but not least, the large-scale nature of the infrastructure must be taken into account, and preserved, at each level of the software archicture.

In this paper, we present the resource management system of a novel grid middleware infrastructure for telecom-oriented applications mixing both computational-intensive aspects and interative, multimedia features. We join [9] in the belief that the specificities of such grid platform are best matched by introducing a multi-agent based approach. The distinctive trait of our approach is to implement multi-agent systems and high-level agent-based negotiation protocols to decide and regulate the use (allocation, reconfiguration, release) of resources within the grid infrastructure. We propose a cooperative ressource management architecture generic enough to support highly heterogeneous applications: no assumptions are done about their execution mode (batch *vs.* non-batch) or requirements. Moreover, relying on a distributed population of heterogeneous, yet cooperative, agents, helps with scalability and dependability issues (e.g. no single point of failure) as well as with heterogeneity and extensibility issues: the diversity of applications requirements can be gracefully mapped onto the heterogeneity of agents. The apparition of new class of requirements leads to the introduction of additionnal new class of agents into the platform.

The outline of the paper is as follows. Section 2 presents related work, followed by a presentation of our specific context in Section 3. Section 4 describes our architecture and a simple case study. Conclusions and perspectives appear in Section 5.

2 Related Work

As stated in the introduction, resource management systems explicitly targeted at interactive grids are scarce. Recent projects include [13], which proposes an abstract agent-based architecture to enforce Service Level Agreements (SLAs) during application runtime. Compared to the work presented here, dynamic allocation is not adressed: the QoS requirements are supposed static and the only enforcement policies envisaged are prioritization mechanisms (or application termination), which are not compatible with the interests of a telecommunication operator. Besides, most integrated management systems in grids exhibit static resource management services, incompatible with our purposes. In most cases, jobs are submitted and exit without the possibility of reallocation in course of execution, as in **Globus** [10], **Sun's Grid Engine** or **Ninf** [15]. As for **Condor** [20], if a participating node is overloaded or becomes unavailable, the Condor daemon takes a snapshot of the job and resumes it on another node. But due to its snapshot algorithm, Condor is suitable only for batch jobs. In **Legion** [16], neither the scheduler nor the scheduling policies are mandated. The framework provides mechanisms to construct ad-hoc schedulers and permits resource specification directly for a run. But the general scheduling architecture supports only one-shot negociations between client and provider and therefore, no dynamic reallocation (same for **Nimrod/G** [5]). Other constraints are the noninterference with the applicative code, which is violated by projects like **NetSolve** [18] (where applications must use one of the APIs provided by the system to perform RPC-like computations) or the automation of administrative tasks, which is not found in **AppLeS** [4] (no tool level scheduling solution provided but instead, an ad-hoc scheduler agent must be developped for each application).

Few projects investigate the dynamic resource allocation on grids. Amidst of them, CoordAgent [11], [17] and ARMS [6]. **CoordAgent** proposes an agent-based solution where mobile agents represent user job requests. The agent encapsulates the code, searches the grid for adequate resources, launches the job, and migrates it to another node in case the current host becomes unavailable. Although the system uses migration for fault-tolerance purposes, it could indeed be used to support dynamic resource allocation for interactive processes, given the possibility to plug-in user/administration policies. In regard to our objectives, the project shortcomings of CoordAgent lie in that it requires modification of the applicative code in order to insert checkpoints and mandates Java or C++. Without alteration of the user code, [17] also proposes to enhance the resource broker by providing it with migration capabilities via reflective techniques. The broker gathers dynamic information about the state of the resource during runtime and reports it to a monitor which in turn computes predictive information. This predictive information is used by an Adapter Manager to make a decision whether job migration is required. The principal drawback to this proposal is the obligation for the applications to be developped in OpenJava, and thus its inability to handle legacy applications. The last project presented here is ARMS. **ARMS** tackles the issues of scalability and adaptability in resource management systems through the use of reconfigurable agents. Each agent represents a grid resource which it manages locally, and cooperates with the other agents to enable resource allocation by exchanging with them service advertisements and discovery requests. A special agent possesses a global view of the system and is capable of modeling and simulating the peer agent's performances during runtime, while optimizing its behaviour (e.g. service advertisement frequency). The system is both scalable (the agents compose a homogeneous society except for the special agent, which does not constitue a single point of failure) and adaptive (through the use of PACE performance prediction toolkit to compute a trade-off between complexity of advertisement and complexity of discovery). But the model relies on assumptions about the application as the attribution of one job per machine, which is relevant for batch applications but not interactive ones, targeted in this work.

3 A Telecom Operator Grid

Our work performs within an telecom operator context. Whereas most grids are targeted toward scientific applications within research center, our platform will have to deal with hosting multiple (possibly a large number of) highly heterogeneous applications, typically ranging from batch-oriented to multimedia and 3-tiers business-oriented applications. Moreover, this platform will be highly distributed (currently multiple sites in France and Japan are involved). To illustrate our proposition, we consider three representative applications: a J2EE application server (hosting applications like devices management), a Batch Queue System used for datamining and videocodecs optimisation applications and a videoconference service. Our goal is to design an architecture that will allow such heterogeneous applications to coexist on a single platform with respect to their QoS and performance needs. A global view of such platform is represented in Figure 1.

3.1 Resources

Physical resources in a telecom-oriented grid include clusters of workstations and servers geographically distributed over long distances, with arbitrary architectures, operating systems and libraries. Each resource is described

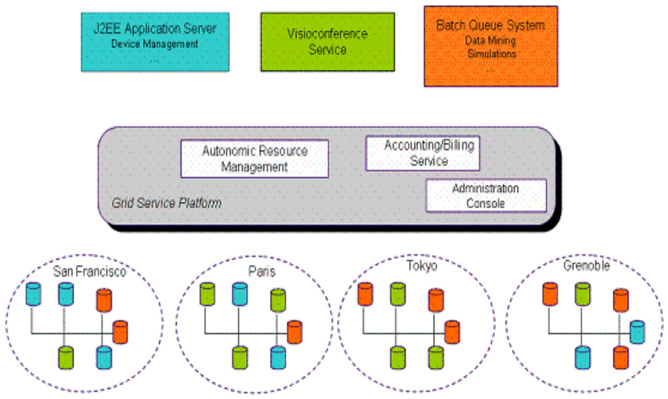

Figure 1. Overview of an enterprise grid platform.

by an XML-based resource profile,[1] providing the necessary semantic markup (architecture, physical resources such as RAM, network properties or storage, operating systems, etc.) to manage it. In such enterprise grid platform, resources are typically owned by the enterprise, but we are considering extending this model toward an open multi-provider context.

3.2 Hosted applications

Since we consider legacy applications, external controllers[2] have to be specifically developped to interface with the underlying platform.

We gather batch applications behind a *Batch Queue Service* that deploys them for execution upon a pool of known resources. Its associated controller has a greedy behaviour as it claims each idle resource available (and suitable) for batch jobs waiting execution. The Batch Queue Service corresponds to the traditional entry point of a data/calcul oriented grid that manages its own resources to handle submitted workflows. We currently use a workflow engine coupled with Condor [20] for such purpose.

The videoconference service represents a legacy application, originally written to be deployed on a dedicated cluster. It balances visioconferences amongst a set of pre-existing conference bridges, hence it needs a *controller* to: (i) monitor the QoS associated with the service and some specific performance/load metrics (typically the number of active videoconferences); (ii) interact with the underlying platform, and in particular negociate additional resources when needed. Those resources will be used to instantiate additionnal conference bridges.

Similarly, the J2EE application server uses additional resources for both performance (load balancing, availability) and dependability (replication) reasons. Resources are typically used to instantiate additional servlet engines, DB servers or even HTTP front-ends. However, it does not exhibit as much real-time constraints as multimedia services.

[1] We do not present any profile samples for sake of space.
[2] Application controllers are further described in Section 4.

4 Architecture for dynamic resources provisionning

Most grid middleware technologies are explicitly designed to deal with batch-oriented applications and thus implicitly rely on this hypothesis for resource management. Conversely, supporting heterogeneous applications requires an application-neutral resource manager. Such resource manager has to support different allocation semantics. We call an *allocation semantic* an allocation process augmented by a specifc set of QoS characteristics. Samples of QoS metrics are: possibility of fault-tolerance by means of application replication or estimated execution time. As an example, whereas a Batch Queue System simply requests any resource matching a given profile, more constrained services can exhibit additional requirements, such as timeouts (*i.e.* a delay after which the resource is not useful anymore), or different semantics, such as atomic co-allocation (*i.e.* allocate a whole set of resources matching some given criteria or none). To capture the heterogeneity of allocation semantics, we adopt a heterogeneous society of software agents. Each allocation policy is implemented as an agent behaviour. To add a novel policy to the platform boils down to enrich it with a new class of agents, without having the need to modify the existing ones. Moreover, ensuring good resource management starts by ensuring a good dependability of the resource allocator, so that resources remain available when faults occur. Hence the use of a distributed set of cooperating *agents* for resource management. Using cooperative distributed agents also helps with scalability and reactiveness issues in such potentially large scale distributed environments. Dealing with heterogeneous applications also implies heterogeneous QoS requirements. To arbitrate amongst the requests, we use service differentiation: services are weighted according to a given utility function (for example, economic utility) and resource allocation is prioritized.

4.1 A generic architecture for resource management

Our architecture deals with resource-centered decision-making. To achieve this purpose, we propose an agent-based middleware supporting arbitrary resource allocation semantics. The following figure 2 gives an overview of the general agent-based architecture. It consists of three layers: (i) an applicative layer with, for each application, a controller responsible for maintaining all metadata associated to the application, monitoring its performance (through

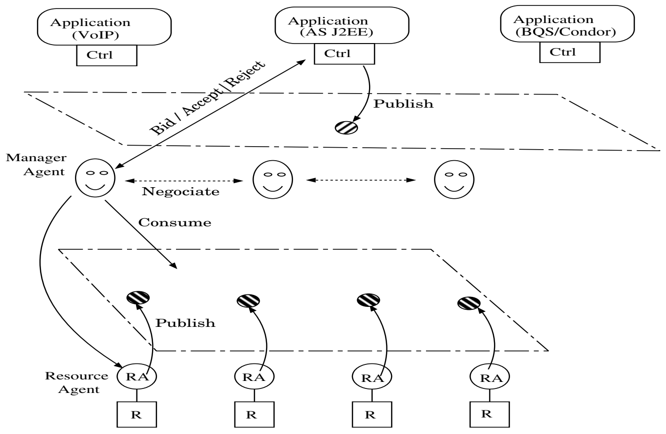

Figure 2. A cooperative resource manager.

QoS metrics) and interfacing application and grid middleware; (ii) the core layer of resource management agents, responsible for resource dissemination, resource discovery and scheduling; (iii) an infrastructure layer consisting of virtualized grid resources and resource agents.

Each resource is handled by a simple agent (called resource agent). Its role is to put in relation for registration purpose the associated resource with an manager agent. To that extent, a shared-space based coordination model is used. A high-level description of the registration process can be stated as follows. Upon activation, the resource agent deposits a request for registration in the shared memory space. It is consumed by the first available manager agent. This agent takes in charge the concerned resource (distinct resources can be managed by the same agent). To enhance resource management dependability, a simple replication schema is applied to managers : each resource possesses a single primary manager (for consistency sake) but knows also of secondary managers or *co-managers*. The designation of the co-managers falls to the manager (they may choose neighbours or clone themselves). Once the co-manager list determined, the manager transmits all the references (both of manager and co-managers) to the resource agent. In turn, the resource agent is responsible for detecting a manager failure and electing a new manager among the co-managers. From an implementation point of view, the choice of a shared memory architecture is affected by scalability issues. As global memory is hardly achievable, we will consider solutions like distributed tuplespaces (e.g. Comet [14]).

Similarly the resource allocation process uses a shared memory space to enable application controllers to request resources from the manager agents. The allocation is done through a *Contract-Net* type protocol. The Contract-Net protocol (CNP) and its extensions is a widely used negotiation protocol within the field of multi-agent systems. Primarily designed for task allocation, it is also perfectly suited to multi-agent resource allocation [7]. It consists of four interaction phases, involving two roles: manager and bidder. The following paragraph describes the first two phases: the announcement phase where the agent initiating the negotiation (i.e. the manager) makes a call for proposal to a number of partners (the bidders), and the bidding phase, where the bidders send their proposals to the manager. The last phases deal wih assignement and confirmation (for more details on the initial protocol see [19], on its specification by FIPA see [2], and on its use in multi-agent resource allocation see [7]). Following the CNP terminology, the application controller stands as the *manager* and the allocation agents as the *bidders*. The idea underpinned by the protocol hereafter described is to exploit the differences in allocation semantics and disposable resources within the population of allocation agents so as to give the application controllers the opportunity to choose between several QoS. The allocation protocol is further detailed.

On the manager side : The *call for proposal* (CFP) mentions resource profiles (hardware or software), the application priority and a *deadline*. The CFP is released through the shared memory space. Upon reception of the proposals, the best suited is selected, based upon application functional and QoS constraints. Contrariwise to a classic CNP, the controller does not wait for the CFP deadline expiration to start evaluating the proposals. If it has received a satisfactory proposal, it can still wait for a better one until the expiration of the CFP deadline (without jeopardizing the anterior proposal). Hence, the choice of a proposal lies on a reactivity / optimality tradeoff.

On the bidder side: The CFPs are addressed in order of priority, so as to enforce service differentiation. Agents compete to bid for a CFP. A bid mentions the resources offered, possible QoS metrics and a deadline. The agents may submit partial bids and perform multiple bids simultaneously. The consequences of this statement are multiple. On the first hand, partial bids have to be processed on the manager-side. Aggregation of partial bids may be a complex application-dependent task. To avoid having to redevelop ad-hoc controllers for each application, we choose to keep controllers as simple and generic as possible. On the counterpart, the resource management middleware is endowed with the task of elaborating satisfying proposals, by means of agent cooperation if necessary. Cooperation protocols through nested CNPs or coalition formation (as described by Aknine et al. [3]), are under study. On the second hand, considering there are several sources of tasks (the application controllers) and that the agents have only limited resources to actually execute the tasks, we face what Schillo et al. in [**?**] call the *Eager Bidder Problem*: the agents have to decide “in how many of the concurrent negotiations they intend to participate and how they should handle their commitments with respect to the local resources at hand”.

At the current stage of our model and for simplicity's sake, we state that agent commitment is definitive (no disengagement allowed until deadline expiration). We keep in mind that further development of the model may challenge this choice.

4.2 Example scenario

To illustrate the allocation processus, let us consider the requests for resources issued by the three previously introduced applications with heterogeneous functional and QoS contraints: a Batch Queue System (BQS), a J2EE application server and a videoconference service.

The BQS manages multiple queues each holding pending jobs. Thus, its associated controller asks for resources[3] matching the requirements of those pending jobs without any order nor relation between those resources. Its utility function maximizes the job throughput, which may lead to select the first proposal satisfying at least one of the resources required by the jobs (first-fit behaviour). The BQS priority is low so the allocation agents will address BQS requests only after having examined the higher priority requests issued simultaneously.

A J2EE application call for proposals, in turn, may express a preference order.[4] Therefore, it may wait until expiration of task announcement deadline for complete proposals or accept an incomplete proposal containing the first resources and consume them while re-emitting a call for the resources left. On the allocation agent side, the agents may bid if they possess required resources satisfying the preference order. They can also cooperate to in order to gather the totality of the resources.

Finally, the visioconference application call for proposal expresses stringent temporal constraints which translate into (potentially short) deadlines. A geographical constraint can enable a selection in agent proposals, based upon physical localization.

4.3 Some issues to tackle

While designing our agent-based cooperative resource manager, we identified three main issues to tackle: the scalability of our multi-agent system and their associated protocols, the starvation problem introduced with service differentiation and the preemption mechanism that is required to enforce service differentiation.

[3]e.g. a Linux PC with 256MB of RAM, two Windows PC with 512 MB RAM and Java 1.5 installed and two Linux PC with 512MB RAM and 1,5GB of storage

[4]That is resources are tied by a preference order: the application may want at least one Linux PC to host an additionnal servlet engine, two if possible, and then it would also like to get a third PC to host a replica of the DB server.

In a decentralized architecture such as the one proposed here, the scalability of the protocols deployed has to be questioned. Sophisticated negotiation protocols may indeed clutter the bandwidth. Previous work [12] has underlined that during multi-agent resource allocation, where agents cooperate through a CNP, execution of the protocol constitutes the main overhead. Performance and scalability of the CNP depend on the size of the system, the number of agents and their load, and the deadline. We plan therefore to take those parameters into account in the allocation policies deployed by our multi-agent system. In case the delay between task execution (minus task length) increases significantly, we plan to reduce the number of messages exchanged by reducing the perimeter of CFPs (for example using metadata on neighbour agents as heuristics), by reducing the access to CFPs according to geographical criteria or by allowing agents to reorganize themselves in more or less centralized configurations (from a flat organization of P2P-type to a hierarchical one).

Priority-based service differentiation may lead allocation agents to neglect applications of lesser priority (typically batch applications), potentially leading to a starvation phenomena. A path to resolve the case is to augment request priority in proportion to age. The prototype in progress explores this solution.

In the context of dynamic allocation, we envision a preemption-based policy: a population of agents is granted the capacity to allocate resources by preempting resources already allocated to applications of lesser priority. This mechanism is meant in particular for multimedia applications which do not suffer delay or interruption. The issues raised by this possibility are numerous, including the redefinition of agent commitment and resource management.

5 Conclusions and perspectives

Grid computing is a promising way for operational costs reduction as well as performance enhancement through a more efficient (flexible and dynamic) use of resources. However, most service-providers and telecommunication companies, such as France Telecom, deal with highly heterogeneous and widely distributed applications. A grid service platform in such a context has to deal with both batch and interactive applications. Not only does static resource provisoning lead to poor resource rationalization, but it also fails to support dynamic QoS requirements. To face such heterogeneous and dynamic context, we propose a generic resource manager able to support arbitrary classes of applications, while avoiding systematic over-provisioning of resource and ensuring service differentiation. This paper introduced the architecture of such a cooperative grid resource manager. It is based on a multi-agent approach to capture the heterogeneity of hosted applications (in terms of allocation semantics) as well as to provide good dependability.

A prototype is currently under development, based on the Jade multi-agent platform [1]. In parallel, the model is being developped along the following lines. We have considered a single owner in the management of our platform resources. Hence the issues of site autonomy and provider concurrence have not been tackled. But our focus on adaptability of resource management policies offers support for the opening of our platform to third-part providers. Another feature we plan to investigate is autonomic grid administration. Grid computing pursues two objectives which are, in order of priority: optimization of application satisfaction rate and optimization of resource utilization rate (objectives which are often contradictory). The present paper has dealt with the first objective; further research shall tackle the second one: given an application satisfaction rate, how can the multi-agent system be re-organized so as to optimize platform-specific performance metrics ?

References

[1] The jade agent framework. `jade.cselt.it/`.

[2] Fipa contract net interaction protocol specification. `www.fipa.org/specs/fipa00029/SC00029H.html`, December 2002.

[3] S. Aknine, S. Pinson, and M. Shakun. A multi-agent coalition formation method based on preference models. *Group Decision and Negotiation*, 13(6):513–538, 2004.

[4] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, S. Spring, A. Su, and D Zagorodnov. Adaptive computing on the grid using apples. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 14(4):369–382, 2003.

[5] R. Buyya, D. Abramson, and J Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In USA IEEE Computer Society Press, editor, *Proceedings of the Fourth International Conference on High Performance Computing in Asia-Pacific Region (HPC ASIA'2000)*, China, 2000.

[6] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd. Arms: an agent-based resource management system for grid computing. *Scientific Programming, Special Issue on Grid Computing*, 10:135–148, 2002.

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

[8] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2003.

[9] I. Foster, N. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other, 2004.

[10] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

[11] M. Fukuda, Y. Tanaka, N. Suzuki, L. F. Bic, and S. Kobayashi. A mobile-agent-based pc grid. In *Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services (AMS'03)*, pages 142–150, 2003.

[12] Z. Juhasz and P. Paul. Scalability analysis of the contract net protocol. In *in Proceedings of Second IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, 2002.

[13] R. Kumar, V. Talwar, and S. Basu. A resource mangement framework for interactive grids. In *Concurrency and Computation: Practive and Experience*, volume 16, pages 489–501. "John Wiley & Sons, Ltd.", 2004.

[14] Z. Li and M. Parashar. Comet: A scalable coordination space for decentralized distributed environments. In *in Proceedings of Second International Workshop on Hot Topics in P2P Systems*, 2005.

[15] H. Nakada, Y. Tanaka, S. Matsuoka, and S. Sekiguchi. The design and implementation of a fault-tolerant RPC system: Ninf-c. In *Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region (HPC Asia 2004)*, pages 9–18, July 2004.

[16] A. Natrajan, M. Humphrey, and A. Grimshaw. Capacity and capability computing in legion. International Conference on Computational Science, May 2001.

[17] A Othman, P Dew, K Djemame, and I Gourlay. Toward an interactive grid adaptive resource broker. In S J, editor, *Proceedings of the Second UK All Hands e-Science Meeting 2003 (EPSRC'03)*, 2003.

[18] K. Seymour, A. Yarkhan, S. Agrawal, and J. Dongarra. Netsolve: Grid enabling scientific computing environments. In *Grid Computig and New Frontiers of High Performance Processing*, volume 14 of *Advances in Parallel Computing*. Elsevier, 2005.

[19] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12), 1980.

[20] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor – a distributed job scheduler. In Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*. MIT Press, October 2001.