Quality Assured Ad Hoc Grids

Kaizar Amin
Argonne National Laboratory, U.S.A.
University of North Texas, U.S.A.

Gregor von Laszewski
Argonne National Laboratory, U.S.A.
Corresponding Author: gregor@mcs.anl.gov

Armin R. Mikler
University of North Texas, U.S.A.

Abstract

This paper presents an integrated architecture for ad hoc Grids developed within the Java CoG Kit project. It provides an overview of the key component frameworks that collectively build the ad hoc Grid architecture. Further, it outlines a formal model that can be formally evaluated. The paper also presents an enhancement to the Java CoG Kit to address requirements posed by ad hoc Grids. It integrates into the Java CoG Kit commodity technologies such as Jxta, and ClassAds.

1. Introduction

Ad hoc Grids are a significant evolution of the traditional Grid computing approach that focuses on the spontaneity of Grid establishment and collaborations [3, 17, 21, 23]. Irrespective of their geographic distributions and organizational affiliations, participants of an ad hoc Grid can form dynamic collaborations on-the-fly without requiring any pre-established environments or policies. Ad hoc Grids differ from traditional Grids in their assumptions for trust-relationship, control-management, and technology-support. The Traditional Grid approach and its architectures enable wide-scale resource sharing between mutually collaborating participants. A pre-established central Grid management authority is responsible for defining the Grid policies, selecting a Grid technology [5, 14], installing Grid services, granting usage authorization, and monitoring the Grid usage. In contrast, ad hoc Grids offer a dynamically defined, loosely coupled, autonomous, and self-organizing Grid environment. Rather than supporting a central management authority, ad hoc Grids adhere to a distributed community controlled Grid environment where participating members are responsible for managing and controlling its own membership policies and services to grant membership. Any participant can join the ad hoc Grid as long as the community grants appropriate membership privileges. Thus, ad hoc Grids facilitate collaborations between "all" potentially interested participants, including mutually collaborative entities that may not be members of virtual organizations as is the case in traditional Grids. A participant can join the ad hoc Grid as a service consumer or provider, thereby also referred as an ad hoc Grid peer. A service providing peer secures its own services and manages the access and usage permissions for its contributed services [22]. Therefore, an ad hoc Grid can also be classified as a peer-to-peer (p2p) Grid that specifically focuses on the establishment of on-the-fly collaborations. We note that not all p2p Grids can be termed as ad hoc Grids. For example, SETI@Home [9] is a p2p Grid. However, it is not an ad hoc Grid since it is neither established on-the-fly, nor does it adopt a distributed community-based control.

Since ad hoc Grids are required to operate in dynamic and transient environments, it can be justifiably argued that they have to deal with the sporadic nature [21] of behavioral patterns defined by the infrastructure and their participants. Ad hoc Grid services are not maintained with a goal of global optimization. Hence, it is difficult to predict and analyze the availability, reliability, and quality of participating services. For example, consider a transaction between a service consumer and a service provider. Since the provider does not adhere to any global optimization policies, the withdraw from the transaction at a time of its convenience can cause consumer tasks to fail. Such a lack of reliability in randomly available services prohibit ad hoc Grids to become a mainstream technology like its traditional counterpart with established quality of service agreements. In order to make the service availability more predictable and improve the reliability of ad hoc Grids, it is important for peers to employ also a quality assurance mechanisms. Quality of service (QoS) models for service reservation and quality specification assists the service consumers to make deterministic assessments regarding the behavior of ad hoc Grids. Further, being an open framework, it is

possible for an ad hoc Grid to host several services offering redundant Grid functionality to enable overprovisioning. In such scenarios, it is important to have a mechanism that allows the service consumer to scope these services based on their quality offerings and consume the services that best matches their requirements. Hence, an ad hoc Grid with appropriate quality augmentations provides an open, adaptive, and self-managing environment without compromising on the availability and reliability.

This paper presents a quality augmented ad hoc Grid framework developed within the Java CoG Kit project [19, 20]. It outlines an overall design of the ad hoc Grid framework and an implementation of the discussed design. The rest of this paper is organized as follows. Section 2 describes the overall architecture design of the Java CoG Kit ad hoc Grid framework. It also outlines a formal model identifying the desired functionality of key ad hoc Grid elements. Section 3 provides a detailed discussion of an ad hoc Grid implementation. Section 4 summarizes the paper and identifies future research issues.

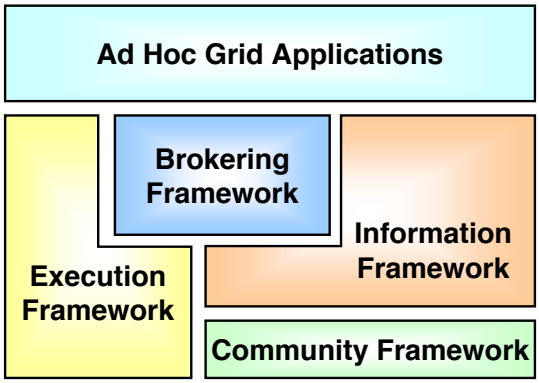

Figure 1. The integrated Java CoG Kit ad hoc Grid framework

2. Architecture Design

Figure 1 shows the high level component design of our architecture. Rather than adopting a tightly integrated architecture, we provide a loose coupling of several frameworks to realize an ad hoc Grid architecture. The loose coupling of component frameworks allows us to replace any of the frameworks with different implementations and technologies without affecting the overall functionality and semantics of the ad hoc Grid. To support the technology- and implementation-independence, we provide a formal abstraction model of our architecture. The ad hoc Grid architecture assumes a service-oriented environment, where resources are contributed and consumed as services. However, no assumptions are made with respect to the technology or implementation details of these services. In the rest of this section, we discuss the role and responsibility of each of the component frameworks outlining important definitions and significant assumptions.

Community management framework The community management framework is a foundation framework that provides a base for the ad hoc Grid peers to establish Grid communities on-the-fly. Essential functionality required from this framework is to allow peers to form an ad hoc Grid, advertise the existence of an ad hoc Grid, discover other existing ad hoc Grids, and acquire membership to an ad hoc Grid.

Information management framework The information management framework is another core component of the ad hoc Grid architecture. An efficient information management system is fundamental not only to ad hoc Grids, but to any traditional Grid system. The information management system represents a distributed service registry where peers that contribute services to the Grid can register their service description and availability. Likewise, consumer peers can extract the description and availability information from the distributed registry. Ad hoc Grids may host several services belonging to the same functional class. For example, an ad hoc Grid might host several job execution services or several data storage services. Under such circumstances, a mere description of the functional characteristics of the service does not differentiate it from other services in the same class. However, a service differentiation scheme is important for providers to better advertise and market their services. Likewise, it is important for the consumers to select target services that best satisfies their requirements. Our service differentiation scheme is based on the quality assurances given by a service provider. Thus, it allows service consumers to make fine-grained service selections as per their quality requirements. A service description is composed of its functional and quality descriptions. While the functional description of a service is static, its quality offerings change dynamically. Hence, the information management framework allows service providers to regularly update the system with an up-to-date snapshot of their quality offerings.

Formally, we denote f as a generic functional class. Example functional classes include job execution, data storage, file transfer, and workflow. We define a consumer task $t(f^{'}, q^{'})$ with functionality requirements $f^{'}$ of a functional class f and quality requirements $q^{'}$. We also define an ad hoc Grid service $s(f^{''}, q^{''})$ supporting functional class f with functional capability $f^{''}$ and quality offering $q^{''}$. An information service $S = \bigcup_{i=1}^{n} s_i$, is a collection of all the

available services in the ad hoc Grid. Let $S_{f'} = \bigcup_{i=1}^{m} s_i$ be a set of Grid services registered by the information service, where $S_{f'} \subseteq S$ and for all $s_i(f_i^{''}, q_i^{''}) \in S_{f'}, f^{'} \leq f^{''}$. Here, $\leq$ represents the "satisfies" relationship. Hence, $f^{'} \leq f^{''}$ denotes that functional requirements $f^{'}$ are satisfied by the functional capabilities $f^{''}$. Likewise, $S_{q'} = \bigcup_{i=1}^{o} s_i$ is a set of Grid services registered by the information service, where $S_{q'} \subseteq S$ and for all $s_i(f_i^{''}, q_i^{''}) \in S_{q'}, q^{'} \leq q^{''}$.

Brokering framework Using the information management framework along with the functional and quality scoping of services, consumers can manually select services to accomplish their tasks. Performing manual task-to-service binding, even in isolated tasks with simple quality requirements can consume considerable amounts of resources. Sophisticated execution flows with complex control dependencies and quality requirements make it impractical to manually perform the service binding for each task. Therefore, to streamline the task execution and service invocation process we integrate an autonomous service brokering framework. A service broker is an autonomous agent local to every consumer. The service consumer delegates all its task execution to its local broker which is responsible for matching the task to the most appropriate service available in the Grid. If $t(f^{'}, q^{'})$ is a task to be executed on the Grid, the service broker is defined as $B = \{t \equiv s_j, \text{ where } s_j \in \{S_{f'} \cap S_{q'}\}\}$. Here, $\equiv$ denotes a successful match relationship. We further expand our brokering model to make intelligent service selections in scenarios where a task can be successfully matched to multiple candidate services. We know that $\{S_{f'} \cap S_{q'}\}$ represents the set of services that can be successfully matched to a task $t(f^{'}, q^{'})$. Let this set of candidate services be denoted as $S_{candidate} = \{S_{f'} \cap S_{q'}\}$. Rather than matching the task with any randomly selected service $s_j \in S_{candidate}$, the brokering framework makes a qualified service selection. We augment the definition of a successful match between a task and a service from $t \equiv s_j$ to $t \equiv^{m_j} s_j$, where m denotes the goodness or quality of the match. The policy for evaluating the goodness of the match can either be predefined by the user or extracted dynamically from every task based on the proximity between the quality requirements of the task and quality offerings of the service. In the event of multiple successful matches, the broker adopts the following logic: $t \equiv^{m_j} s_j$, where $s_j \in S_{candidate}$ and $m_j = |M|_{max}$ for $M = \bigcup_{i=1}^{n} m_i$. With the extended brokering model, we further enhance the quality of adhoc Grid interactions because the tasks not only bind to a service that satisfies its needs, but they bind to a service that "best" satisfies its needs.

Execution management framework The execution framework is responsible for providing a dynamic plug-n-play invocation mechanism allowing consumers to utilize services offered by the providers. An ad hoc Grid imposes no restrictions on the use of any Grid technology [5, 7, 14, 16]. Peers can contribute services supporting any functional class and may be implemented in a technology of their convenience. Intuitively, consumers invoking services implemented in a specific technology would also be required to support that technology and formulate their tasks specification in context to that technology. However, it is impractical for the consumers to keep updating their task requirements and specification dependent of the service implementation adopted by the providers. Closely binding the consumer task specification with the service implementation provides a rigid, inflexible, and non-scalable architecture. We decouple the task specification process from the service implementation- and technology-dependent details by adopting an abstraction-based "task-translator" execution paradigm [1, 2]. The functional requirements $f^{'}$ of any ad hoc Grid task t represents an abstract set of requirements independent of any technology-specific details. For example, if the consumer task is to execute some remote job, its functional requirements $f^{'}$ can be supplied as the following abstract specification: executable name, executable path, input arguments, environmental variables, and input files. We note that this specification contains no implementation-specific details such as the "execution universe" in case of Condor [14] jobs or "gass server" details in the case of Globus [5] jobs.

In order to provide a formal model of the execution framework, we extend our previous definition of an ad hoc Grid service from $s(f^{''}, q^{''})$ to $s(f^{''}, q^{''}, i)$ where i denotes the service implementation technology. An implementation translator $p(i, f)$ is a client-side component to invoke any service $s(f^{''}, q^{''}, i)$ of functional class f. For example, a service supporting the remote job execution class ($f_{execution}$) implemented in Globus v4 (gt4) technology is represented as $s(f^{''}_{execution}, q^{''}, gt4)$. An implementation translator for this service, represented as $p(gt4, f_{execution})$ is a client component that is capable of understanding the protocol-specific semantics of the GT4 job execution service. Thus, given a task $t(f^{'}, q^{'})$ and service $s(f^{''}, q^{''}, i)$, the translator extracts the technology-independent functional specification $f^{'}$ and translates it into implementation i specific constructs understood by the corresponding service s to invoke $f^{''}$. A consumer x hosts a set of implementation translators $P_x = \bigcup_{i=1}^{n} p_i$. A consumer is capable of invoking a service $s(f^{''}, q^{''}, i) \iff \exists p(i, f)$ and $p(i, f) \in P_x$.

Hence, the execution framework supports any service with arbitrary functional class implemented in any technology provided there exists a suitable implementation translator for it. Every service description points to an appropri-

ate Internet location to retrieve the corresponding translator. During the invocation of a service, the consumer extracts the translator from the recommended location if it is not already available in the set P hosted on the consumer.

Reservation framework The brokering framework offers a mechanism to bind a task t to some service $s \in \{S_{f''} \cap S_{q''}\}$ such that the quality requirements are satisfied by the offerings. However, even though the service is capable of offering a specific level of quality $q^{''}$, the actual quality provided to the consumer may not necessarily be the same. Let $q_p^{''}$ be the actual quality provided to the consumer by the service s with a maximum quality offering $q^{''}$. It may be possible for a set of independent tasks $T = \bigcup_{i=1}^{n} t_i$ belonging to different consumers to invoke a service $s(f^{''}, q^{''}, i)$ at the same time. If every task $t_i(f_i^{'}, q_i^{'}) \in T$ has the same priority, they are each provided a quality $q_p^{''} = \frac{q^{''}}{n}$. Since $q_p^{''} < q^{''}$, it may be possible that $q^{'} \not\leq q_p^{''}$, where $\not\leq$ implies the "does not satisfy" relationship. Such undeterministic quality provisions dependent on the synchronous usage pattern of ad hoc peers reduces the overall reliability and quality of ad hoc Grids. Therefore, to improve the reliability of the quality of a service and to make it independent of the service demand, we improve the execution model to explicitly extract quality provisioning assurances prior to the service invocation. In other words, a provider makes a "deterministic guarantee" to the consumer with respect to the value of $q_p^{''}$ prior to its invocation. These guarantees are implemented and offered by a suitable reservation scheme. A consumer can reserve a service for itself giving that consumer exclusive usage privileges for the duration of the reservation.

We augment our previous definition of a task from $t(f^{'}, q^{'})$ to $t(f^{'}, q^{'}, r)$, where r represents a finite time-based reservation for exclusive service usage. A service reservation r can be further classified as an "elastic" reservation r_e or a "fixed" reservation r_f. An elastic reservation simply specifies the duration of the requested service usage whereas the fixed service reservation specifies the exact start date and end date of the service usage. Thus, if the consumer wishes to invoke a service for a given duration independent of the starting date, it specifies its requirements as an elastic reservation. On the other hand, if the service usage is tightly bound to some specific date, a fixed reservation should be used. Prior to invoking a service $s(f^{''}, q^{''}, i)$, the implementation translator $p(i, f)$ is responsible for negotiating the usage rights of the service as per the reservation requirements r of the task. For an elastic reservation, the service provider has the flexibility to decide an appropriate start time for the service invocation by the consumer. On the other hand, in a fixed reservation requirement the start time for the service invocation is determined by the consumer. If both peers are in agreement with the service usage parameters, they form a reservation agreement, also known as a service level agreement. A reservation agreement is an enforceable obligation on the part of the service provider to ensure that the consumer receives the expected service quality, $q_p^{''} = q^{''}$. Hence, it can be seen that the discussed ad hoc Grid architecture offers enhanced quality assurances at different levels of Grid interactions including the information framework, the brokering framework, and the execution framework. As discussed earlier, such deterministic quality assurances allows Grid applications to focus on dynamic and adaptive application requirements without any compromise in Grid availability, reliability, and quality.

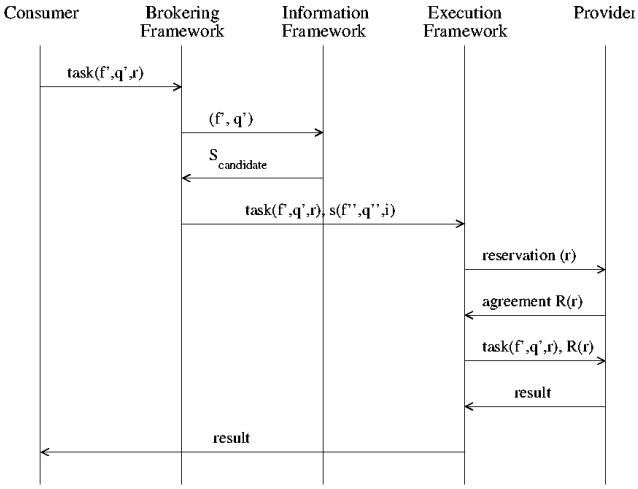

Figure 2. Communication semantics between the different frameworks of the CoG ad hoc Grid architecture

Figure 2 summarizes the inter-framework communication semantics. The consumer formulates a task $t(f^{'}, q^{'}, r)$ and delegates it to the brokering framework. The brokering framework interact with the information management framework to get $S_{candidate}$ for the task. Based on its policy, the brokering framework selects the best service $s \in S_{candidate}$ to match the task t. The task is bound to the selected service and delegated to the execution framework. The execution framework negotiates the reservation agreement with the service provider. On reaching a mutually-acceptable agreement, the execution framework invokes the service within the scope of the agreement. The result of the service invocation is appropriately delegated back to the consumer.

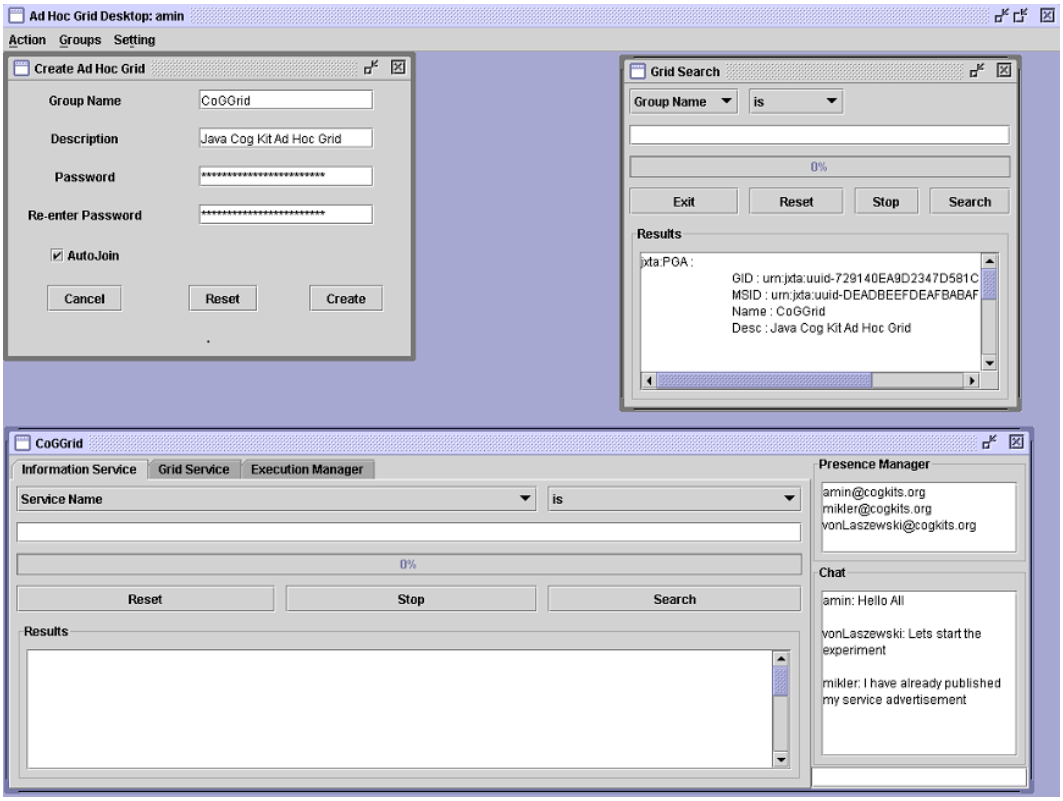

Figure 3. User interfaces for ad hoc Grid creation, discovery, presence management, and Grid-wide communication

3. Implementation

In order to validate the formal model discussed in Section 2 we have implemented an ad hoc Grid prototype. Being an integral part of the Java CoG Kit project, the implementation adheres to the concept of providing advanced ad hoc Grid solutions reusing existing commodity technologies. More importantly it follows the design based on abstractions that are augmented with providers [17]. In this section we describe the implementation details and the technologies used for each of the component frameworks depicted in Figure 1.

The community management framework provides functionality to create ad hoc Grids, discover existing Grids, join discovered Grids, and communicate with peers of an ad hoc Grid. However, an implementation of this framework is required to operate in a distributed environment independent of any participant entity. Several ad hoc, distributed, and self-organizing community management frameworks have been proposed in literature [4, 6, 8]. Significant research has also been conducted in the area of distributed peer-to-peer lookup and discovery mechanisms [11, 13]. Any of these frameworks or algorithms can be adopted for our prototype. However, we build on top of the capabilities offered by project Jxta [8], which is at this time the most comprehensive framework available. Jxta is a collection of open peer-to-peer protocols and services that allow any device with a "network heartbeat" to communicate and collaborate with other Jxta peers autonomously. It provides a mechanism to create virtual ad hoc collaborations without exposing any of the underlying peer-to-peer protocol complexities. It enables the formation of a self-organizing super-peer-based overlay network on the Internet. Further, to adapt to the ever-changing participation of peers in the community, Jxta employs a completely decentralized advertisement and discovery of peers and services using advanced techniques such as distributed hash tables [15].

Jxta technology is primarily based on the concept of "peer groups". A Jxta peer can seamlessly create, discover, and join peer groups within the Jxta environment. It can also exchange application-pertinent information within the scope of a joined group. For the ad hoc Grid implementation, we provide a one-to-one mapping between the Jxta group and an ad hoc Grid community. Using Jxta protocols, every participating member becomes the member of the "Global Grid Group". Such a global Grid community

provides the requisite infrastructure to advertise the availability of an ad hoc Grid and discover the existence of other Grids on the network.

Even though the a peer may discover existing production Grids, it cannot participate in that Grid without formally acquiring membership to that production Grid. Based on their adopted policy, every ad hoc Grid implements its own group admission-control scheme including voting, passwords, certificates, and web of trust [12]. Our prototype supports open Grid memberships and password-based Grid memberships.

It further enhances the collaborative experience by providing utility communication tools for presence management and Grid-wide user messaging. Figure 3 shows the user interface for the ad hoc Grid creation, Grid discovery, presence management and group communication.

Listing 1. Quality offerings of a service

```
[
  Name = ''org.cog.execution.service'';
  Type = ''Provider'';
  Class = ''Execution'';
  OS = ''Linux'';
  Arch = ''Intel'';
  KFlops = 21893;
  MemoryGB = 2;
  Requirements =
    Other.Class == ''Execution'';
  Rank = Self.MemoryGB-Other.MemoryGB;
]
```

We assume that every service belongs to some functional class and that all services belonging to a given functional class exhibit the same functionality description. For example, all services belonging to the job execution functional class can execute remote jobs when provided with appropriate parameters such as executable name, arguments, environment variables, input files, and output files. Likewise, all tasks for the same functional class have a standardized abstract template to express their functionality requirements, also called the task specification. Quality descriptions, however, are not bound to a functional class. Irrespective of the associated functional class, peers can express their quality descriptions in terms of arbitrary parameters suiting their needs. To provide an elegant quality description language that is expressive for peers to depict their quality characteristics yet formal to allow its unambiguous evaluation, we use the ClassAd (Classified Advertisement) language [10].

The ClassAd language is a functional language with its basic unit being an expression. A ClassAd is a set of name/-value pairs where every value may be an arbitrarily complex expression, including nested record expression and list of expression. The ClassAd language is a comprehensive system in itself and its complete analysis is beyond the scope of this paper. For a deeper understanding of its semantics, the reader is directed to [10]. Listing 1 shows a sample quality description from a provider describing the quality offering of a service named "org.cog.execution.service". It shows the various quality attributes provided by this service such as flops, available memory, supporting operating system, and hardware architecture. It also contains a "Requirements" attribute expressing the quality requirements of this service. Further, the "Rank" attribute specifies the provider policy in evaluating the goodness of tasks it is willing to process. In this example, it implies that tasks with lower memory requirements get better ranking. Listing 2 shows the sample quality requirements from a service consumer highlighting its requirements and ranking policy.

Listing 2. Quality description of a consumer

```
[
  Name = ''org.cog.execution.task'';
  Type = ''Consumer'';
  Class = ''Execution'';
  MemoryGB = 0.05;
  Requirements =
    Other.Class == ''Execution'' &&
    Self.MemoryGB < Other.MemoryGB;
  Rank = Other.KFlops;
]
```

Listing 3. Service advertisement from a provider

```
ServiceAdvertisement :
PeerName : amin@mcs.anl.gov
ServiceName : org.cog.execution.service
Class : execution
ServiceContact : hot.anl.gov:2119
ServiceTechnology : GT2
Keywords : Java CoG Kit
Description : Java CoG Kit GT2 service
ClassAd : [
  OS = Linux;
  Arch = Intel;
  KFlops = 21893;
  AvailDisk = 13000;
  AvailMemory = 1000;
  Requirements = (AvailMemory > 10000);
  Rank = (1/Other.ReqMemory)
]
```

Every contributed service has an associated advertisement. This advertisement encapsulates all relevant information of that service such as its name, functional class,

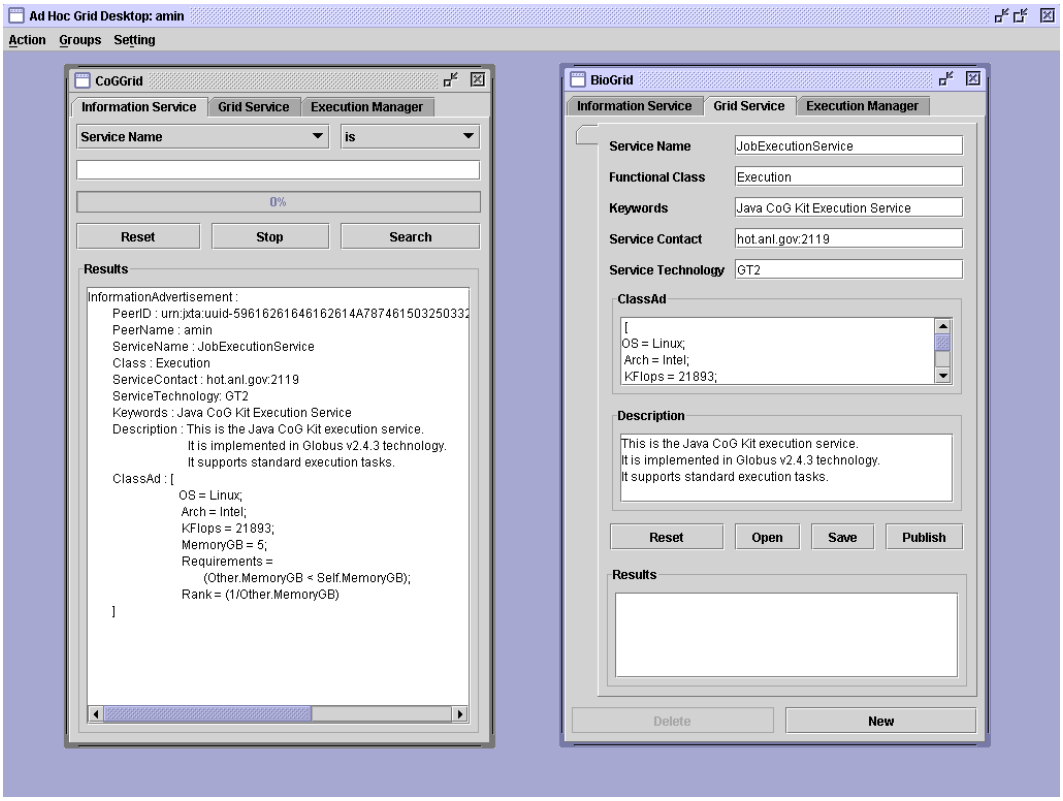

Figure 4. User interfaces for publishing service advertisements and discovering published advertisements

service contact, functional description, quality description (ClassAd), and an implementation technology. The service contact represents the service endpoint to be used for communication with the service. Listing 3 shows a service advertisement from a Globus Toolkit v2.4 (GT2) job execution service.

The distributed information service allows providers to advertise their service descriptions and enable consumers to discover existing services. The information service is an extension of the core discovery service provided by project Jxta. The Jxta discovery service enables an infrastructure to publish and discover arbitrary advertisements using loosely coupled rendezvous nodes and distributed hash tables [15]. A rendezvous node in the Jxta framework is a special peer that maintains a cache of published advertisements. When a peer joins the ad hoc Grid, it automatically seeks a rendezvous peer in that community. If no rendezvous peer is found, the new peer itself becomes the rendezvous peer for that community. If its connection with the rendezvous peer fails or if the rendezvous peer itself fails, the peer actively seeks connection with another rendezvous peer. When member peers wish to publish an advertisement, they push it to their rendezvous peers. The rendezvous peer caches this advertisement and may further push it to additional rendezvous peers (as a fault tolerance mechanism) selected by the calculation of a hash of the advertisement. Peers that wish to discover advertisements, query their corresponding rendezvous peer. If the rendezvous peer contains the queried advertisement in its cache, it honors the request. Otherwise, it will contact other known rendezvous peers in the community to fulfill the request. The information service extends the discovery service to exclusively publish and discover "service" advertisements. It also enables a mechanism to periodically update the rendezvous cache with up-to-date service advertisements. Such a scheme not only assists in maintaining an updated snapshot of service capabilities but also helps in flushing advertisements of inactive service from the rendezvous cache. Figure 4 shows the user interfaces to publish service advertisements and discover published advertisements.

We support two modes of task submissions. In the first mode, referred to as *direct-mode*, the consumer knows the details of the service it wants to invoke including the service contact, functional class, and technology implementation.

The consumer provides the task specification with the relevant service details. The consumer is allowed to make

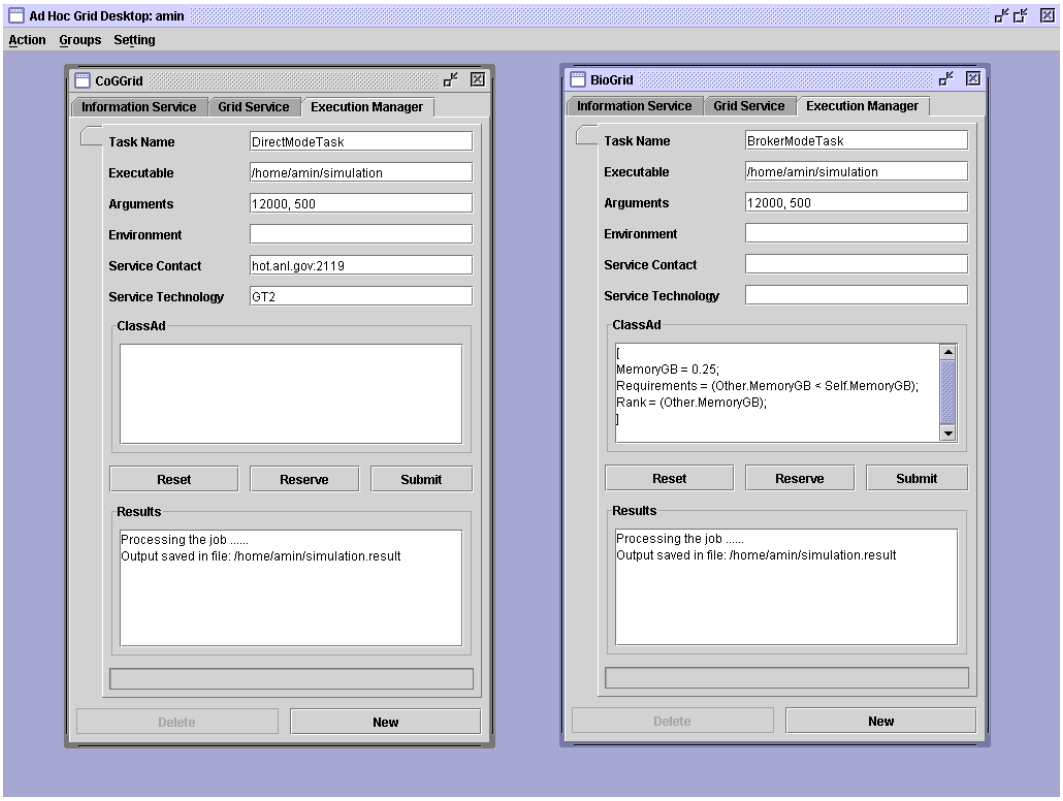

Figure 5. User interfaces for submitting tasks in direct-mode and broker-mode

immediate and advanced service reservations (if supported by the service) before submitting the task requirements. Such reservations guarantee exclusive access to the service during the reserved period. The second mode of task submission, referred as the *broker-mode*, is a more flexible mode of task submission allowing the service broker to perform the task-to-service binding.

The consumer only provides the generic task specification and its quality requirements (ClassAd) without any knowledge of the target service. The broker, interacts with the distributed information service and extracts a list of services whose quality offerings (ClassAd) match the quality requirements (ClassAd) of the task. Two ClassAds are said to match each other if the "Requirements" attribute in both ClassAds evaluate to *true* in the context of the other ClassAd. Further, the brokering component grades every task-to-service match based on the "Rank" criteria specified in the task ClassAd. A successful match with the best ranking is handed to the execution framework for remote service invocation. Figure 5 shows the user interfaces for task submission in direct mode and broker mode whereas Figure 6 shows the user interface to make time-bound service reservation.

The implementation of the execution framework adopts an abstraction-based service invocation mechanism offered by the Java CoG Kit [1, 2]. The execution framework is supplied with an abstract specification of the task along with service-specific details. The execution framework appropriately deploys the technology translators (also called protocol providers) to invoke the given service. A detailed discussion of all the constructs involved in the execution framework is beyond the scope of this paper. For a better understanding of an abstraction-based Grid execution framework we refer to [2].

4. Summary and Future Work

In this paper we describe a novel architecture for ad hoc Grids that integrates a suite of component frameworks. The discussed architecture combines a community management framework, information management framework, brokering framework, and an abstraction-based execution framework. We present the formal design of the ad hoc Grid architecture as well as provide details of an implementation, developed as an integral part of the Java CoG Kit. The design and implementation of our architecture focuses on the issues of community-control, quality assurances, and spontaneity of service contribution and invocation. The implementation uses several commodity technologies including Java, Java

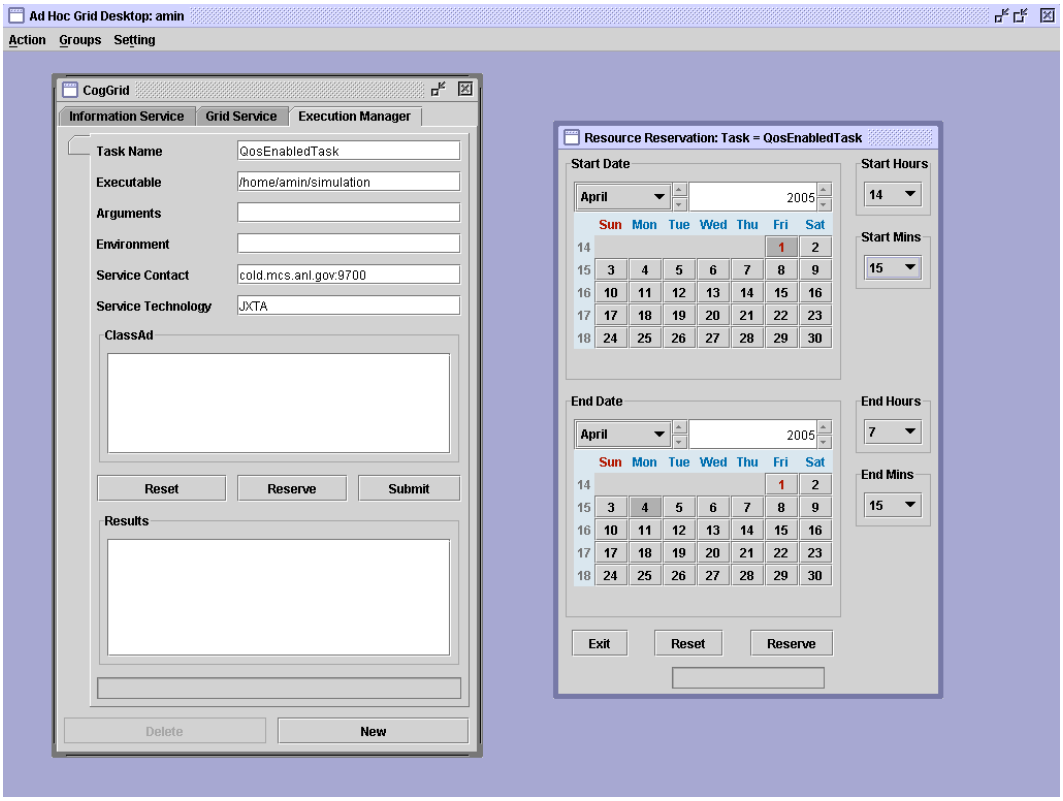

Figure 6. User interface for making immediate and advanced service reservation

CoG Kit, Jxta, and ClassAds.

Ongoing research is focusing on integrating an autonomous, robust, and flexible authentication and authorization control framework within our architecture. It is important to enhance the security of ad hoc Grids with a policy-based security infrastructure that can be conveniently managed by the individual participants without any central control. Additional research activities are focusing on integrating a trust and reputation framework [18] with existing quality provision schemes. Such enhancements will allow the community to collectively isolate malicious peers that do not fulfill their quality promises.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus Project research and development. The Java CoG Kit Project is supported by DOE MICS, and NSF Alliance.

References

[1] K. Amin, M. Hategan, G. von Laszewski, and N. J. Zaluzec. Abstracting the Grid. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2004)*, pages 250–257, La Coruña, Spain, 11-13 Feb. 2004.

[2] K. Amin, G. von Laszewski, R. A. Ali, O. Rana, and D. Walker. An Abstraction Model for a Grid Execution Framework. *Euromicro Journal of Systems Architecture*, 2005. Accepted for publication.

[3] K. Amin, G. von Laszewski, and A. R. Mikler. Toward an Architecture for Ad Hoc Grids. In *12th International Conference on Advanced Computing and Communications (ADCOM 2004)*, Ahmedabad Gujarat, India, 15-18 Dec. 2004.

[4] Berkeley Open Infrastructure for Network Computing. Web Page.

[5] The Globus Project. Web Page.

[6] Gnutella Homepage. Web Page.

[7] A. S. Grimshaw and W. A. Wulf. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1):39–45, January 1997.

[8] Project JXTA. Web Page.

[9] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. SETI@home-massively distributed computing for SETI. *Computing in Science & Engineering*, 3(1):78–83, January–February 2001.

[10] R. Raman. *Matchmaking Frameworks for Distributed Resource Management*. PhD thesis, The University of Wisconsin-Madison, 2000.

[11] S. Ratnaswamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Applications, technologies, architectures, and protocols for computer communications*, number ISBN:1-58113-411-8, pages 161–172, San Diego, CA, August 2001. ACM SIGCOMM.

[12] N. Saxena, G. Tsudik, and J. H. Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 104–113. ACM Press, 2003.

[13] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Applications, technologies, architectures, and protocols for computer communications*, number ISBN:1-58113-411-8, pages 149–160, San Diego, CA, August 2001. ACM SIGCOMM.

[14] D. Thain, T. Tannenbaum, and M. Linvy. *Grid Computing: Making the Global Infrastructure a Reality*, chapter Condor and the Grid, pages 299–336. Number ISBN:0-470-85319-0. John Wiley, 2003.

[15] B. Traversat, M. Abdelaziz, and E. Pouyoul. A Loosely-Consistent DHT Rendezvous Walker. Technical report, Sun Microsystems, Inc, March 2003.

[16] Unicore. Web Page.

[17] G. von Laszewski. The Grid-Idea and Its Evolution. *to be published.*, 2005. Argonne National Laboratory, Argonne, IL 60439, U.S.A.

[18] G. von Laszewski, B. Alunkal, and I. Veljkovic. Toward Reputable Grids. *(to be published)*, 2004.

[19] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[20] G. von Laszewski, J. Gawor, S. Krishnan, and K. Jackson. *Grid Computing: Making the Global Infrastructure a Reality*, chapter Commodity Grid Kits - Middleware for Building Grid Computing Environments, pages 639–656. Communications Networking and Distributed Systems. Wiley, 2003.

[21] G. von Laszewski, J. Gawor, C. J. Peña, and I. Foster. InfoGram: A Peer-to-Peer Information and Job Submission Service. In *Proceedings of the 11th Symposium on High Performance Distributed Computing*, pages 333–342, Edinbrough, U.K., 24-26 July 2002.

[22] G. von Laszewski and M. Sosonkin. A Grid Certificate Authority for Community and Ad-hoc Grids. In *7th International Workshop on Java for Parallel and Distributed Computing, published in the Proceedings of the 19th International Parallel and Distributed Processing Symposium*, Denver, CO, 4-8 Apr. 2005. IEEE.

[23] Y. Wang, F. D. Carlo, D. Mancini, I. McNulty, B. Tieman, J. Bresnahan, I. Foster, J. Insley, P. Lane, G. von Laszewski, C. Kesselman, M.-H. Su, and M. Thiebaux. A High-Throughput X-Ray Microtomography System at the Advanced Photon Source. *Review of Scientific Instruments*, 72(4):2062–2068, Apr. 2001.