Available online at www.sciencedirect.com
SCIENCE DIRECT®
Web Semantics: Science, Services and Agents on the World Wide Web 4 (2006) 116–123
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

On agents and grids: Creating the fabric for a new generation of distributed intelligent systems

Yolanda Gil

Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292, USA

Received 10 October 2005; accepted 27 March 2006

Abstract

The semantic grid is the result of semantic web and grid researchers building bridges in recognition of the shared vision and research agenda of both fields. This paper builds on prior experiences with both agents and grids to illustrate the benefits of bringing agents into the mix. Because semantic grids represent and reason about knowledge declaratively, additional capabilities typical of agents are then possible including learning, planning, self-repair, memory organization, meta-reasoning, and task-level coordination. These capabilities would turn semantic grids into *cognitive grids*. Only a convergence of these technologies will provide the ingredients to create the fabric for a new generation of distributed intelligent systems.

Keywords: Semantic grid; Semantic web services; Distributed AI; Multi-agent architectures

1. Introduction

Semantic grid researchers (www.semanticgrid.org) have noted the common themes in the semantic web and grid research agendas, as represented in technical publications in both areas [1–4]. These commonalities have been articulated in [5–8]. Not surprisingly, in recent years both communities share an interest in the widespread service-based architectures. Semantic web researchers are developing extensions to web services to include semantic representations (http://www.w3.org/2002/ws/swsig). Grid services and the more recent WS framework (http://www.globus.org/wsrf) align web services with more traditional features of grids. With this emphasis in service-based paradigms, these fields are now sharing more than ever a research agenda with yet another community: autonomous agents and multi-agent systems. Agent systems have historically been developed in a modular fashion, with a clear advertisement to other agents of their capabilities, which is an important emphasis of service-based architectures. The research agenda in the agents community [9,10] shares much with the semantic web and grid research agendas. Some commonalities have been pointed out in [11,9,12], including trust, negotiation, resource allocation, composition, and autonomy.

This paper argues that neither grid nor agent systems alone can deliver on their promise without building on one another. The arguments put forward are based on first-hand experiences with agent systems [13,14], scientific workflow creation and execution in grids [15–24], and metadata catalogs for grids [25–27]. In our work, we have used grid techniques to address shortcomings in agent systems, and AI techniques to bring not only semantics but also intelligent reasoning capabilities to grids.

The paper is organized in two main sections, one focused on why agent systems need grids, and the second on why grids need agent systems. The first section discusses capabilities that grid technologies provide that would address important limitations of agent systems, concentrating on providing robust and sustainable performance. These facilities include state, persistence, and lifecycle management. The second part of the paper discusses important capabilities developed and used in the agents community at large that would address current limitations of grid computing. These capabilities include learning, planning, interaction, and coordination. We argue that these capabilities would take grids to a new level of scale and sophistication, able to make complex informed decisions and flexibly adapt their performance to new information and unexpected situations.

2. Why grid: limitations of current multi-agent systems for robust and sustainable performance

Robustness is a very important requirement for distributed problem solving of the kind performed by multi-agent systems. Robustness is challenging when heterogeneous collections

E-mail address: gil@isi.edu.

1570-8268/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2006.03.002

of components need to be coordinated in a highly dynamic non-centralized execution environment. These are the environments that grids were designed to address. Reliability, quality of service, and robustness have been central themes in grid research as essential architectural principles. Research in multi-agent systems has focused more on distributed operations at the task level, concentrating on architectures, agent communication languages, and coordination [51–53]. We argue that grid environments provide an ideal substrate for developing multi-agent architectures and distributed problem solving capabilities. This section describes key features provided by grid services [28,29] and their relevance to support robust implementations of agent-based systems.

Our experience with multi-agent systems, and in particular with the Electric Elves project [13], can be used to motivate these issues. Electric Elves supported users with office tasks and integrated a number of agents that were heterogeneous in their function and in their implementation, and were deployed in a distributed environment. Some of the agents accessed web information sources, others matched requests with agent capabilities, and others communicated with users about task status. As we strived to use the system 24/7 inside and outside of our organization, we discovered that a non-trivial amount of effort was required to keep the groups of agents functioning appropriately. The capabilities of each agent and the kids of information they exchanged were clearly specified. Yet, even such an agent community with modest distributed and heterogeneous nature posed clear challenges. As the agents operate, it was necessary to monitor their status and the status of specific requests. This was done manually, but ideally it should handled automatically. The kinds of status questions that came up included: (1) Is the other agent still working on my request? (2) Does the agent still need my reply? (3) Has the agent already responded but the response was not received? (4) What is the status and progress of my request? Other problems came up when machines were down or connections failed. Should the requesting agents have a memory of their requests and resend them, or should the servicing agent keep track of their requests and reinstate them upon restart? Who should notice that an agent is not running and who should restart it? Although it was possible to manage the multi-agent system manually, this is not a desirable option and clearly impossible if multi-agent systems are to scale up in number and heterogeneity.

These challenges are not uncommon and it is reasonable to consider they hinder growth and scale of deployed agent communities. In a published study [30] noted problems in agent infrastructures with significant development effort in dependability and survivability at larger scale (hundreds of distributed agents). We believe that these are symptoms of the need for better distributed infrastructure to build multi-agent systems.

These issues can be summarized as follows:

1. Introspection: Once a request is sent to an agent, it would be useful to be able to query the agent regarding the receipt of the request, the status of the request, and perhaps any intermediate results that the agent may have generated. It would be useful to setup automatic notifications of changes in a request status.
2. Tracking multiple requests: For each agent, we have to implement a mechanism for tracking multiple requests. For each request, the agent has to maintain its status, current result, etc. and detect the same request sent repeatedly when the message acknowledging its receipt delayed.
3. Persistence: Each agent can implement a database to store requests so they can be recovered in case the agent process dies. Special purpose scripts may be needed to check the agent process and restart it if it is not running.
4. Shelf life of requests: Each agent needs to implement a mechanism to deal with expiration of requests. Otherwise, all requests would be active forever. Other services that may not have a clear date associated with them may be harder to handle in terms of terminating the request.
5. Evolution of the specifications: The agent specification (in many cases due to changes in the underlying web site) may change over time. This requires manually tracking changes in other agent specifications, so that request message formats could be updated.

The implementation of individual agents can be extended to support these questions in an ad hoc manner. However, these problems clearly raise the need for better infrastructure to support robust continuous operation as tasks are executed in a dynamic distributed environment. Instead of developing ad hoc solutions for these problems, it is useful to investigate whether we can build on existing approaches in distributed computing that are designed to address these very issues.

2.1. Some useful concepts in grids

We conducted a study of how grid services would support a more robust agent infrastructure, focusing on how the requirements discussed above would be addressed in grids. We implemented a set of agents using grid service definitions and found that the built-in mechanisms in grid services provided facilities to support much of the functionality we needed regarding the state of the agents and the requests. A detailed account of the implementation is provided in [14]. This section describes some useful concepts in grids illustrated in the context of our multi-agent system. Our implementation was done with the Open Grid Services Infrastructure (OGSI) [29] and its GT3 specification. Here we use terminology from that work. Its new generation is the WS-Resource Framework (WSRF) (http://www.globus.org/wsrf) and has slightly different terminology but the same underlying core concepts. See [31] for a detailed mapping of terminologies.

Grid Services are extensions of web services that incorporate several key features that are crucial for robust distributed operations. One key feature of grid services is the notion of lifecycle management: a request has a lifetime that is managed by the system by assigning it a process. A request sent to a grid service may create a transient process called *grid service instance* with a unique identifier, a *handle*, that can be used to locate and query the instance. The service instance exists only for a limited amount of time, after which the instance will be destroyed. If required, the client/application can also extend the expiration