Agent Societies

Magnitude and Duration

Michael N. Huhns • *University of South Carolina* • *huhns@sc.edu*

If you only need agents to search the Web for cheap CDs, scalability is not an issue. The Web can support numerous agents if each acts independently. In short order, however, billions of embedded agents that sense their environment and interact with us and other agents will fill our world, making the human environment friendlier and more efficient. These agents will need not only scalable infrastructures and communication services, but also scalable social services encompassing ethics and laws. Research projects are underway around the world to develop and deploy such services.

In this installment of "Agents on the Web," I'll take a look at the critical relationship between scalability and intelligent agents.

On the Horizon

Here are some interesting agent applications where scalability plays a fundamental role:

- A large HMO is considering deploying agents to represent its patients and members. Most current medical information systems assist either medical providers, such as physicians and nurses, or medical administrators, such as hospitals and insurers. Few try to assist patients directly—there are just too many of them. After all, a city might have a handful of hospitals and clinics, a few thousand doctors, but millions of patients. Agents looking out for the welfare of individuals could generate reminders and alerts when patients don't fill their prescriptions, schedule medical appointments and procedures, or simply inform people of new treatments relevant to their condition. The basic goal is to improve people's medical outcome by helping them before and after they enter the health-care system, not just while they are in it.[1]
- An express-mail service embeds a small processor with RF capabilities and an agent in each package it handles. The agent helps track the package and, more importantly, negotiates with other agents to make sure its package gets the service and attention it deserves. The service handles millions of packages each month.
- Similarly, agents represent logistics items in military deployments. Each item has an agent-based smart card containing a mechanism for communicating locally and globally; a reasoning engine; a knowledge base with information about routes, conveyances, and conflict-resolution strategies; and the agent's objectives, priorities, needs, and relationships to other items. An agent will complain if its item is put in a railroad car heading in the wrong direction or on an unrefrigerated truck when it is perishable. By negotiating with other agents, an agent representing a crate of machine guns will make sure the appropriate boxes of ammunition get on the same truck going to the right destination.[2]
- At online stores, such as Amazon and Barnes & Noble, and at online auctions, such as eBay, agents represent customers. They help personalize customer services by comparing notes with other agents to decide which products a customer is most likely to want and to make sure that buying and selling follow proper and efficient procedures. Along the same lines, Stanford's LIRA research system uses collaborative filtering to find the things liked by people (represented by their agents) who are similar to you (represented by your agent);[3] Yenta, from the MIT Media Lab, performs matchmaking by clustering the agents with similar interests;[4] and ReferralWeb at AT&T locates people with needed expertise.[5]

What do the agent systems in all of these scenarios have in common? Agents represent and act on behalf of real-world entities, from inanimate objects to people. Also, the systems are too complicated to have a centralized architecture, so a distributed-agent architecture seems to be the only reasonable approach. Finally, successful implementations

 1089-7801/02/$17.00 ©2002 IEEE

require that the systems scale up to a large number of agents and interactions.

Notions of Scalability

Fundamentally, scalability is the ratio between performance and resources. We can think about an agent system's scalability in terms of its characteristics when applied to a domain that changes in size, or in terms of how the agent system achieves scalability.

An agent-based system can cope with a growing application domain by increasing the number of agents, each agent's capability, the computational resources available to each agent, or the infrastructure services needed by the agents to make them more productive.

Alternatively, an agent-based system can exhibit scalability one of three ways:

- As the amount of resources available to a fixed system of agents increases, system performance should increase.
- As the number of agents in a system increases to match increases in the number of patients, packages, or entities in a domain, the system should continue to function as designed.
- A scalable system should perform *better* by taking advantage of the additional capabilities offered by the increased number of agents.

Scalability can also be dynamic or static. Static systems must be recompiled and restarted when the number of agents or the resources available to them change. Dynamic systems can accommodate changes in agents and resources during runtime. Obviously, dynamic systems are preferable.

Scalability in Practice

Scalability is not a problem for reactive agents, because they do not use any system resources until they receive a message. Increasing the number of a system's reactive agents simply causes a storage problem for the agents and a possible communication bottleneck for the messages they exchange.

Proactive, *deliberative*, agents consume resources as long as they exist. They evaluate their current circumstances and then plan their actions to achieve both immediate and long-term goals. They are continuously active, where even deciding to do nothing requires active deliberation and, thus, resources.

Physically, you can achieve scaling as follows:

- *Distribute* system components—the agents and the service providers—across multiple physical machines, using distributed computing technologies such as DCOM, CORBA, Java RMI, .NET, and JINI. Unfortunately, this approach can introduce communication latencies.
- *Replicate* the components on multiple physical machines, using distributed computing technologies similar to those in the distributive approach. Unfortunately, this tactic introduces consistency problems among the multiple copies, which limits its use to applications that are mostly static (a lot of communication is required to restore consistency when systems change).
- *Schedule* the components intelligently to execute only when and for as long as necessary to optimize the use of available resources. You can also arrange the components into hierarchies or other organizational structures to make their interactions more efficient.

Each of these techniques has served successfully to deploy large systems of agents.

Resources

Distributed Interactive Engineering Toolbox: http://www.ens-lyon.fr/~despres/DIET/index.htm

The Grid: http://coabs.globalinfotek.com

Jess: http://herzberg.ca.sandia.gov/jess/main.html

Yenta: http://foner.www.media.mit.edu/people/foner/yenta-brief.html

Zeus: http://www.labs.bt.com/projects/agents/zeus/index.htm

Scaling Infrastructure Services for Agents

The services agents require—services provided by agents to each other and by the infrastructure—must also scale.[6] Agent services include name services, location services, directories, facilitators, and brokers. Infrastructure services include message transports, human interfaces, and CPU cycles.

The Internet, though DNS, already has an established means for scalable name services, which agent-based systems can use. DNS essentially scales through replication.

Scaling directory services are more problematic. A directory service, such as LDAP, consists of attribute-value pairs that an agent can search for matches to its requirements, much as a person searches through a yellow pages. In general, an agent might need to exhaustively search an entire directory for each look-up. An index can shorten the search time, but such indexes are difficult to maintain in a distributed setting.

Scalability Experiments

In investigating the effect of communication on scalability, researchers developing the ZEUS multiagent framework discovered that the maximum communication load grows, at worst, linearly with the number of agents.[7]

A research team at the University of Saskatchewan used the DICE framework to investigate the computational load of creating and executing 1,000 simple agents on a set of 10 remote hosts.[8] The results demonstrated the feasibility of moving agents to less

busy hosts for load balancing and also that response times remain reasonable—a few seconds at most—for agents that need to respond to people. Other results with complex rule-based agents (agents that incorporated the JESS reasoning engine) showed that 400 agents could execute acceptably within the same computational framework.[8]

The DIET framework uses lightweight threads and thread-management techniques to enable more than 100,000 simple agents to execute on a single host machine.[9]

In a different kind of experiment, a team at the University of South Carolina is investigating the scalability of a system of medium-complexity, heterogeneous agents. The agents form geometric shapes on a 2D grid by communicating with nearby agents. Although only 60 agents are involved, 60 different people constructed them.[10] For online reputation assessment experiments in (human) social networks, the team is scaling the system to more than 500 agents.

In a similar effort for scaling heterogeneous agents in a distributed and dynamic world, the DARPA Control of Agent-Based Systems (CoABS) program has developed an infrastructure called the Grid (see http://coabs.globalinfotek.com). The Grid has integrated agents and components from more than 20 independent projects and has operated successfully in a series of naval fleet battle exercises and other applications, from information retrieval to military command and control. Built using Sun's Jini services, the Grid can integrate agent-based systems, object-based applications, and legacy systems. Agents in the Grid communicate point-to-point, so communications scalability (to the limit imposed by network bandwidth) is not a problem. The Grid relies on a lookup service for registration and discovery that is centralized, which is a potential bottleneck. However, recent experiments with up to 10,000 agents show that registration and discovery are essentially independent of the number of agents registered.[11]

Long-Lived, Adaptable Agents

Scalability applies not only to the number of agents and their interactions, but also to agent lifetimes and interaction duration. Most agents in use today are designed for short lives in relatively static online worlds. For example, an agent might be programmed to access the Web pages of five online stores and find the best price for a given music CD. While this is underway, the sites are presumed to be static and, when finished, the agent dies.

Future agents—especially those who represent users in their dealings with a ubiquitous computing world—must live for many years.

In contrast, future agents—especially those who represent users in their dealings with a ubiquitous computing world—must live for many years. Such agents will learn and adapt as they and their users encounter new situations, making it impractical for them to be recreated from scratch. Their needed infrastructure services must also be designed to exist for many years. They will also need new kinds of services: social services to help them cooperate in solving larger tasks, and legal services to help them meet their obligations, and ensure their rights.

Acknowledgements

The National Science Foundation supported this work under grant number IIS-0083362.

References

1. J. Davis, M. Huhns, and R. Bonnell, "Using Objects and Patterns for Building Compliance Agents in Healthcare," *Proc. OOPSLA'98 Midyear Workshop OOT for Insurance and Health Care*, 1998, pp. XX-XX
2. R. Staats, L. Glicoes, and M. Huhns, "A Multiagent Environment for Department of Defense Distribution," *Adaption and Learning in Multiagent Systems*, G. Weiss and S. Sen, eds., Springer-Verlag Lecture Notes in Artificial Intelligence 1042, Berlin, 1996, pp. 53–84.
3. M. Balabanovic and Y. Shoham, "Learning Information Retrieval Agents: Experiments with Automated Web Browsing," *Proc. AAAI Spring Symp. Information Gathering from Heterogeneous, Distributed Resources*, AAAI Press, Menlo Park, Calif., 1995, pp. 13–18.
4. L. Foner, "Yenta: A Multi-Agent, Referral Based Matchmaking System," *Proc. First Int'l Conf. Autonomous Agents (Agents '97)*, ACM Press, New York, 1997. pp. XX-XX
5. H. Kautz, B. Selman, and M. Shah, "ReferralWeb: Combining Social Networks and Collaborative Filtering," *Comm. ACM*, Vol. 40, No. 3, 1997, pp. 63–65.
6. F. Brazier, M. van Steen, and N. Wijngaards, "On MAS Scalability," *Agents'01 Workshop on Infrastructure and Scalability for Agents*, ACM Press, New York, 2001, pp. 121–126.
7. P. De Wilde, H.S. Nwana, and L.C. Lee, "Stability, Fairness, and Scalability of Multi-Agent Systems," *International Journal of Knowledge-Based Intelligent Engineering Systems*, Vol. 3, No. 2, 1999.
8. R. Deters, "Scalability and Multi-Agent Systems," *Agents'01 Workshop on Infrastructure and Scalability for Agents*, ACM Press, New York, 2001; www.cs.cf.ac.uk/User/O.F.Rana/agensts2001/papers/02_deters.pdf.
9. P. Marrow, "Scalability in Multi-Agent Systems: The DIET Project," *Agents'01 Workshop on Infrastructure and Scalability for Agents*, ACM Press, New York, 2001; www.cs.cf.ac.uk/User/O.F.Rana/agents2001/papers/18_howden.pdf.
10. V. Holderfield, *A Foundational Analysis of Software Robustness Using Redundant Agent Decision-Making*, tech report, Center for Information Technology, Univ. of South Carolina, Columbia, Nov. 2001.
11. M.L. Kahn and C. Della Torre Cicalese, "CoABS Grid Scalability Experiments," *Agents'01 Workshop on Infrastructure and Scalability for Agents*, ACM Press, New York, 2001; www.cs.cf.ac.uk/User/O.F.Rana/agenst2001/papers/04_kahn_et_al.pdf.

Michael N. Huhns is a professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology.