2002

Weaving a Computing Fabric

Michael N. Huhns
University of South Carolina - Columbia, huhns@sc.edu

Larry M. Stevens
University of South Carolina - Columbia, stephens@cec.sc.edu

John W. Keele

Jim E. Wray

Warren M. Snelling

See next page for additional authors

Publication Info

Published in *IEEE Internet Computing*, Volume 6, Issue 5, 2002, pages 88-91.
http://ieeexplore.ieee.org/servlet/opac?punumber=4236

Author(s)
Michael N. Huhns, Larry M. Stevens, John W. Keele, Jim E. Wray, Warren M. Snelling, Greg P. Harhay, and Randy R. Bradley

Weaving a Computing Fabric

Michael N. Huhns and Larry M. Stephens • *University of South Carolina*
John W. Keele, Jim E. Wray, Warren M. Snelling, Greg P. Harhay, and Randy R. Bradley • *Meat Animal Research Center, USDA*

Agents — in the form of elves, sprites, avatars, jinis, and genome agents — are forming the new fabric for our computing infrastructure. Working behind the scenes, they are automating mundane tasks to free people for more interesting and creative endeavors.

Because they are easy for people to use, agents perform this role well. People work best with things that seem familiar. Software components are typically not familiar, but if the components resemble assistants, aides, helpers, or colleagues, users will feel more in control and be more productive.

Agents seem to be the right size as well — larger than methods, which are often too numerous to find and use, but smaller than major applications, which are often too complex to be completely understood. (Does anyone understand *all* the features in MS Excel?) Agents also enable scalability. An agent-based system can cope with a growing application domain by increasing the number of agents, each agent's capability, or the computational resources available to each agent.

Given their flexibility and all-around suitability for this broad range of tasks, it is not surprising to see the variety of agent-based infrastructures on the landscape.

Electric Elves

Researchers at the University of Southern California Information Sciences Institute have developed an agent system that represents people in planning and scheduling tasks.[1] The agents, termed *electric elves*, run autonomously once deployed, continuously seeking opportunities for the person they represent to attend meetings. (For their technology's next generation, let's hope they deploy anti-elves to help people *avoid* meetings!)

The elves are flexible and have a wide range of capabilities: they can arrange a lunch meeting for project participants, including the selection of food, or keep track of a project visitor's schedule, including possible arrival delays and the associated necessity for rescheduling meetings. Because they are implemented on a platform of PDA and Global Positioning System devices, elves can even keep track of the participants' physical locations.

Elves interact with each other often. In scheduling a meeting, for example, one elf might delegate the task of finding a room to another. To facilitate these interactions, the system employs an agent matchmaker system, using a declarative language to represent each elf's capabilities and requests.

Jini

Researchers at Sun have developed the agent-like Jini system (www.jini.org) for controlling access to physical devices that might be computerized within our homes, offices, and environment. Jini extends Java from one machine to a network of machines. It uses remote method invocation (RMI) to move code around and provides mechanisms for devices, services, and users to join and detach from the network. This approach is essentially a model for Web services within a local-area network, which might be implemented using the Bluetooth or IEEE 802.11x protocol.

The Jini Web services infrastructure offers several advantages:

- transactions support a two-phase commit protocol,
- clients lease services for specific durations,
- lookup services can be arranged hierarchically, and
- services occupy nodes in tuple spaces, called JavaSpaces.

 1089-7801/02/$17.00 ©2002 IEEE

Some disadvantages are:

- lookup services require an exact match on the name of a Java class (or its subclass) and
- clients and servers exchange code synchronously via RMI.

Successful Web service developments and deployments require that services be described so that requestors can accurately find and use them. Early on, multiagent systems confronted this problem when attempting to solve problems cooperatively or compete intelligently. Although no general, formal language for describing tasks has emerged, all descriptions must include information about the problem to be solved, the nature of the expected solution, the resources involved or required, and a time by which a solution is needed. Optional information includes the task's importance or priority, the solution's value, and the other requestors that might be interested in the task's performance.

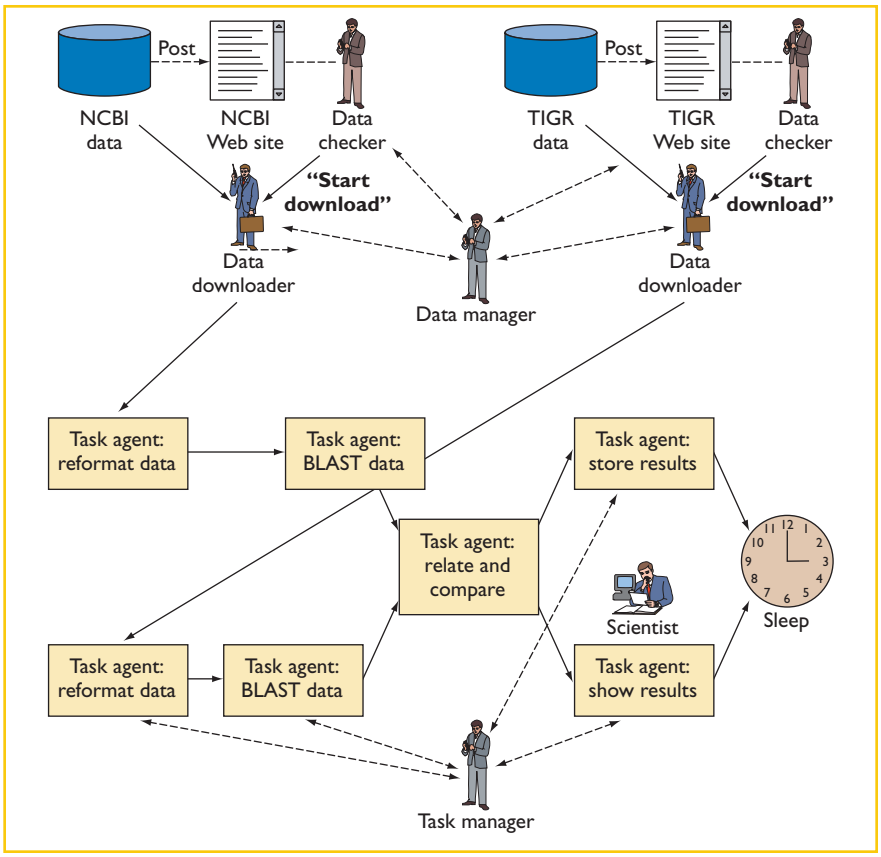

Figure 1. Operational view of agent-based infrastructure for managing genomics information. A data manager agent creates and monitors agents that download new information when it becomes available. A task manager agent then manages the agents that process the new information.

Genome Agents

Spurred by the great potential for improving the health and welfare of people and animals, the excitement surrounding the decoding of the human genome has fueled an explosion in the amount of information available about genomics. In addition, much of the progress sprang from advances in algorithms for manipulating genomics information and sharing the results over the Internet. Further progress will be aided by assisting end users — biologists and geneticists — in obtaining and using the latest and best information.

Maintaining a comparative genomics information system is challenging because data resources are distributed, heterogeneous, and dynamic. Numerous analysis programs are available for sequencing data to determine gene structures and functions, but to stay current, an effective system must download information from external resources frequently, reconcile differences in data format, apply various analysis algorithms, and integrate the resultant information.

The operational requirements for the system we constructed for use at the US Department of Agriculture, essentially mandate the use of an agent-based infrastructure because multiagent software systems can integrate information from many external resources and reconcile format differences.[2]

- The system should be robust, so, if it crashes, restarts occur automatically or with minimal operator intervention without any loss of work.
- The system should be controllable and modifiable from multiple locations.
- System monitoring should be possible from multiple locations.
- The system's workload should be distributable across multiple platforms.
- Several administrators or users should be able to initiate, modify, or halt data-acquisition activities.
- System control should be declarative and not procedural: Users should be able to tell the system *what* they would like done, and the system should figure out *how* to do it.
- The system should be modifiable while it is running, without requiring a restart for the modifications to take effect.
- The system should not do any unnecessary work, such as downloading an already current file.

As Figure 1 illustrates, our autonomous multiagent system can download sequence information from a variety of online data sources, apply the Basic Local Alignment Search Tool (BLAST) to the information to find matches, infer sequence function from annotations, integrate the results into comparative maps, and display the maps

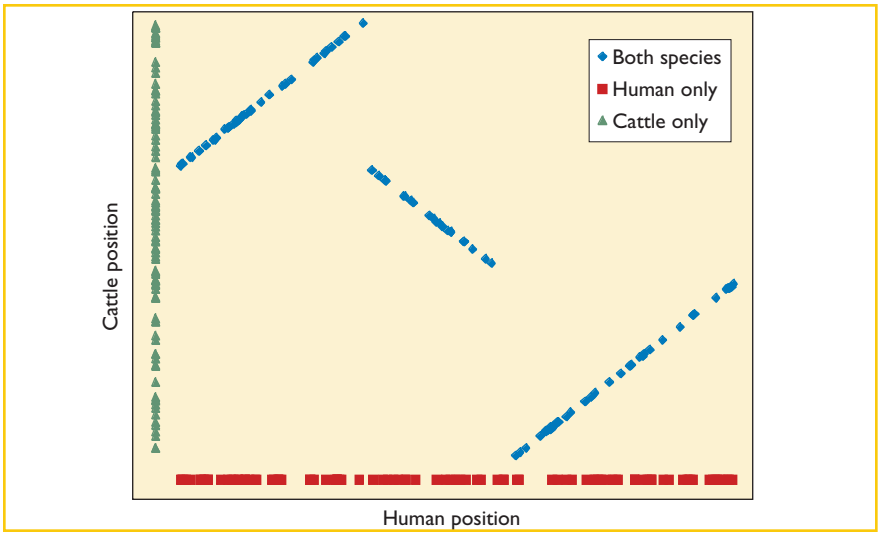

Figure 2. A comparative map of human versus cattle genomic information. The map shows where on the cattle genome (vertical axis) the corresponding human genetic information (horizontal axis) are located. Here we see that human and cattle have mostly the same genetic information, but it is arranged differently.

and data.[3] The agent-based system features a mediated general architecture[4] that contains a management agent, directory agents (for both yellow and white pages), a database control agent, agents responsible for monitoring Internet sources and downloading new data, and a workflow agent to control processing tasks on the new data.

A database control agent is in charge of all runtime changes to a local database. Because all database changes occur via this agent, it can both make the changes and easily inform appropriate agents about the changes. A data-manager agent starts multiple pairs of data-checker–data-downloader agents in response to messages from users or after querying instructions from a database during initialization. Each data-checker agent looks for changes in the file date of the external resource for which it is responsible. When it detects a change, the agent sends a message to its corresponding data-downloader agent to download the file.

Once a download completes, the system sends messages to agents controlling downstream processes — such as gunzip, formatdb, cross_match, BLAST, and perl data parsing — indicating that new data is in the pipeline to be integrated into the information system. Task and downloading agents, perhaps distributed over multiple platforms, interact by exchanging messages encoded in the Foundation for Intelligent Physical Agents agent-communication language syntax; we implemented all agents using the public-domain Java Agent Development Framework (Jade) (http://sharon.cselt.it/projects/jade/). Overall, the system behaves as a distributed agent-based workflow execution system.[5]

Our genomics information system lets researchers produce comparative maps that show how to map

- cattle and swine chromosomal sequences to human chromosomal sequences
- human chromosomal sequences to cattle and swine chromosomal sequences

Figure 2 shows an example result.

Avatars and Sprites

An avatar is a graphical character or persona that inhabits a virtual world, whereas a sprite is an inanimate graphical object in that world. Virtual worlds are the domain of computer games, but they are moving increasingly into online training and education. They help engineers and artists shape artifacts for the real world, assisting us in understanding things we would not normally be able to see, such as atomic, cosmic, and imaginary structures.

Avatars populate these virtual worlds, interacting with each other and the world and often representing our interests and us. By providing a familiar form for what is essentially a computing operation, they let people comfortably interact with complex computations.

Services for an Agent-Based Infrastructure

Elves and avatars cannot exist on their own; they need an environment and an infrastructure in which to live and act. The most essential service an environment can provide is to enable agents to locate and engage each other. Agent-based infrastructures typically provide this capability in the form of a directory service. Other services might provide transaction-processing monitors, loggers for saving records and recovering from failed operations, visualizers to help track the status of ongoing activities, sniffers to eavesdrop on agent conversations, and controllers to start and stop agents. To be understood, agents must communicate according to well-defined semantics, often provided by an ontology service. For example, Figure 3 shows the ontology used for the genomics domain. Development and execution platforms such as Jade provide most of these services.

Implementing Peer-to-Peer Systems

The information technology community is looking hard at peer-to-peer and grid computing architectures (www.gridcomputing.org/grid2002) to provide agents the processing services (CPU cycles) they need. These technologies represent an approach to distributed computing based on the use of idle computer resources. Nodes already recruited into the distributed computing network locate other nodes by executing search algorithms that

ask the questions, "Are you available?" or "Do you have the appropriate data?" If the node is available, it is told what to do. If it doesn't have the right data, it is given the data. The search for available nodes and execution proceed under the basic assumption that each node is stupid and cannot decide for itself what it should do.

This approach works as a minimal default assumption, but it fails to capitalize on each node's potential intelligence, previously cached results, improved algorithms, or other capabilities that could be exploited to improve performance.

A collection of agents or elves, or whatever gremlin comes down the pike next to run on a distributed collection of platforms, can behave like a distributed operating system — or like an *intelligent* P2P system, in that the agents can actively cooperate to accomplish tasks.

Conclusion

As sources of information relevant to a particular domain proliferate, we need a methodology for locating, aggregating, relating, fusing, reconciling, and presenting information to users. Interoperability thus must occur not only among the information, but also among the different software applications that process it. Given the large number of potential sources and applications, interoperability becomes an extremely large problem for which manual solutions are impractical. A combination of software agents and ontologies can supply the necessary methodology for interoperability.

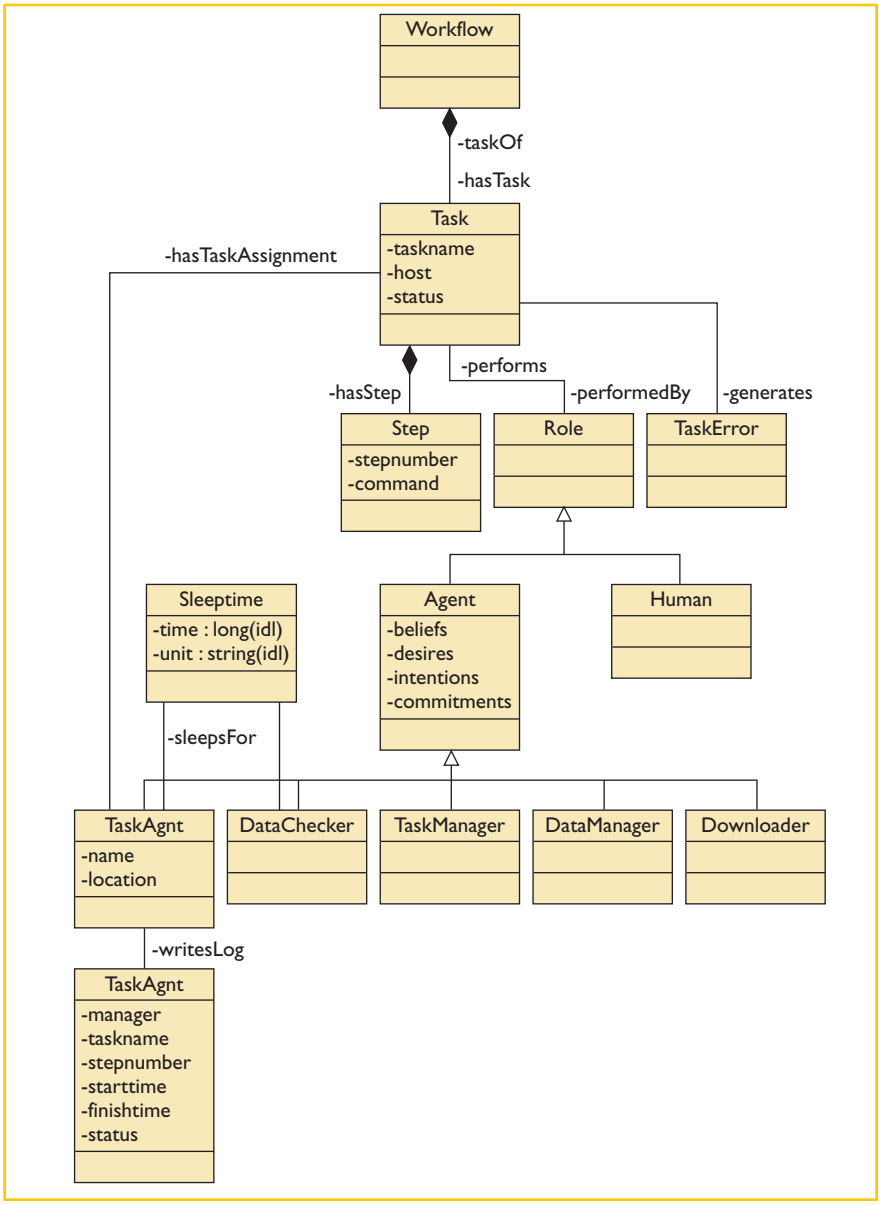

Figure 3. Ontology of genomics information management. Genomic agents use this ontology of concepts to interact and communicate.

Acknowledgment

The US Department of Agriculture supported the work on genomic agents described here.

References

1. H. Chalupsky et al., "Electric Elves: Agent Technology for Supporting Human Organizations," *AI Magazine*, Summer 2002, pp. 11-24.
2. K. Bryson et al., "Applying Agents to Bioinformatics in GeneWeaver," *Cooperative Information Agents IV,* Lecture Notes in Artificial Intelligence, vol. 1860, Springer-Verlag, New York, 2000, pp. 60-71.
3. J.W. Keele et al., "Intelligent Software Agents for Managing Distributed Genomics Data," *Proc. 28th Conf. Int'l Soc. Animal Genetics (ISAG)*, Blackwell Publishing, Oxford, UK, 2002, to appear.
4. M.N. Huhns and M.P. Singh, "All Agents Are Not Created Equal," *IEEE Internet Computing,* vol. 2, no. 3, May/June 1998, pp. 90-92.
5. M.N. Huhns and M.P. Singh, "Managing Heterogeneous Transaction Workflows with Cooperating Agents," *Agent Technology: Foundations, Applications and Markets*, N.R. Jennings and M.J. Wooldridge, eds., Springer-Verlag, Berlin, 1998, pp. 219-240.

Michael N. Huhns and Larry M. Stephens are professors of computer science and engineering at the University of South Carolina, where they conduct research in multiagent systems and ontologies.

John W. Keele, Jim E. Wray, Warren M. Snelling, Gregory P. Harhay, and Randall R. Bradley are researchers in animal genomics and bioinformatics at the Meat Animal Research Center, US Department of Agriculture, Clay Center, Nebraska.

Readers can contact the authors at {Huhns, Stephens}@sc.edu and {Keele,Wray,Snelling, Harhay,Bradley}@email.marc.usda.gov.