Decentralized Resource Control for Multi-Agent Systems

Nadeem Jamali, Indratmo, Xinghui Zhao
Department of Computer Science
University of Saskatchewan
57 Campus Drive
Saskatoon, SK, S7N 5A9, Canada
{n.jamali, j.indratmo, x.zhao} @usask.ca

Gul A. Agha
Department of Computer Science
University of Illinois
201 N. Goodwin Avenue
Urbana, IL 61801, USA
agha@cs.uiuc.edu

Abstract

In an open system, multi-agent computations must compete for resources required for satisfying their goals. We describe CyberOrgs, a hierarchical model for acquisition and control of resources for multi-agent systems in a market of resources. Programming abstractions and constructs are introduced for implementing systems of CyberOrgs. A prototype implementation of the model as an Actor program is described, and scheduling approaches for an efficient implementation are discussed.

1. Introduction

There are multiple sources of uncertainty in a computational environment. An application independent source of uncertainty for a computation emanates from other computations competing for available resources. In an open system where computations may enter or exit the system at any time, this is a typical scenario. Coordinating resource access by agents is hence critical to reduce uncertainty and enable agents to make control decisions for best global performance.

Ether [4] was the first language to address explicit allocation of resource in concurrent systems, in which sponsors were assigned to processes to support their computations. In Quantum [5], computations require *energy* to execute. Computation tasks are contained in hierarchical *groups* which also serve as *tanks* of energy. When a group's computations terminate, its energy is absorbed by its parent group.

2. CyberOrgs

CyberOrg ("Cyber Organization") is a model for resource control in an open multi-agent system. Cyberorgs organize computational resources as a market, and their ownership and control hierarchically. Specifically, each cyberorg encapsulates a concurrent computation and resources available for its execution. Cyberorgs also own *eCash* with which they buy resources from other cyberorgs; however, they may not create eCash *ex nihilo*.

Cyberorgs organize resources as a tree. A cyberorg hosted by another cyberorg receives resources from its host in exchange for eCash payments, according to a pre-negotiated contract.

Even though ownership relationships are represented as a hierarchy, the ownership of a resource is absolute for the duration of an ownership. This means that a cyberorg decides autonomously whether to sell a resource during the interval in which it owns it, for a part of that interval.

Although terms of ownership of resources are decided between sellers and buyers, physical distribution of resources between cyberorgs needs to be controlled at the location of the resource. For example, processor cycles are typically distributed by schedulers. A hierarchy of schedulers independent of the hierarchy of cyberorgs is hence required for distributing processor cycles among cyberorgs or parts of cyberorgs physically located on a single machine.

The CyberOrg model separates concerns of computations from those of the resources required to complete them. We assume that computations are carried out by actors [1], and we represent the resource requirements of each computation by the sequence of resources required to complete it. *Ticks* serve as the unit of a consumable resource such as processor time. Every computation requires a certain number of ticks to complete.

Progress is represented by transitions occurring with introduction of ticks into the system. When a tick is inserted into a cyberorg, it may pass the tick on to a client cyberorg, use it for progressing on its chores or on its actors. Whether a tick is passed on to a client or used locally depends on the contracts that the cyberorg has with its clients.

As illustrated in Figure 1, a new cyberorg is created by using the `isolate` primitive, which collects a set of ac-

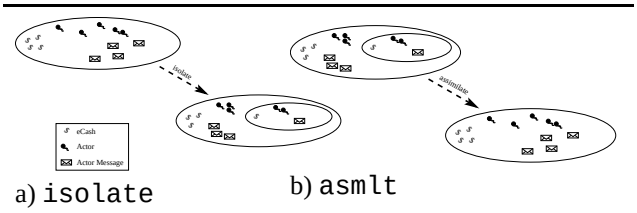

Figure 1. Creation and Absorption

tors, messages, and electronic cash, and creates a new cyberorg hosted locally. A cyberorg disappears by assimilating into its host cyberorg using the `asmlt` primitive, relinquishing control of its contents to its host.

A cyberorg may realize that its resource requirements have exceeded what is available by its contract with the host cyberorg. This triggers its attempt to migrate. The tasks required for a cyberorg to migrate are: search (for a potential host), negotiate (a contract with potential hosts), and migrate (to a selected host).

An operational semantics of CyberOrgs can be found in [3]. Preliminary analysis of the model has found cyberorgs to hold the following properties.

- The amount of eCash in the system remains constant.
- If no new ticks are inserted into the root cyberorg, the system will eventually become dormant.
- If transfer of eCash is limited to purchase of ticks, cyberorgs carrying eCash only migrate up the cyberorg tree, and the price of ticks only reduces in fixed quantums, then the system will eventually become dormant.

An executable specification of CyberOrgs has been developed using the formal specification language Maude [2]. Maude is an implementation of rewriting logic, which is a logic particularly suited for modeling dynamic and concurrent systems because it enables one to define a system's states in modules and to express conditions that trigger state transitions of the system as rewrite laws.

3. Prototype

A prototype CyberOrg implementation has been built using Actor Foundry, a library of Java classes supporting Actor functionality. Actor programs may be written and executed in a run time that supports operational semantics of the Actor model.

A system of cyberorgs is implemented by directly subclassing from `CyberOrg` and `AppActor` classes. There are three important components of a CyberOrg system implementation: cyberorg classes which implement a runtime system for each class of cyberorgs, managing "tick" consumption by local application actors, and securing "tick" resource for a cyberorg from other cyberorgs; application actor classes, which hold the behaviors of application actors; and client and server negotiator classes implementing specific strategies for negotiating contracts.

An important hurdle in efficiently implementing cyberorgs is the model's hierarchical structure. If cyberorgs are implemented naively so that each cyberorg is represented by a scheduler, a system of cyberorgs would be realized by a hierarchy of schedulers. Because scheduling overhead is typically dominated by the cost of thread suspension and resumption, suspension of any non-leaf cyberorg would also require suspension of all cyberorgs in the path from it to the executing actor.

Our approach to implementing cyberorgs is by flattening the schedule at run-time. Instead of launching a new scheduler for each cyberorg, all cyberorgs' internal schedules are merged into a single flat schedule of actors which preserves all processor time allocations. Because the resulting schedule is flat, there is virtually no additional scheduling overhead resulting from the cyberorg infrastructure.

4. Conclusions

Agents sharing an execution environment invariably compete for available resources, possibly in ways impacting global performance of a multi-agent application. CyberOrgs offer a model for acquisition and control of resources for multi-agent applications, which allows applications to execute in an environment of predictable resource availability. The model achieves a separation of concerns by representing resource requirements of an application separately from its functionality. An executable specification of the model has been developed using the Maude specification language. Work is ongoing on developing an efficient implementation of CyberOrgs and studying properties of systems of CyberOrgs.

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, Mass., 1986.

[2] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. A maude tutorial. Technical report, SRI International, Computer Science Lab, 2000.

[3] N. Jamali and G. Agha. Cyberorgs: A model for decentralized resource control in multi-agent systems. In *Proc. of WS on Representation and Approaches for Time-Critical Decent. Res./Role/Task Alloc. at AAMAS 03*, Melbourne, 2003.

[4] W. A. Kornfeld and C. Hewitt. The scientific community metaphor. *IEEE Transactions on System, Man, and Cybernetics*, 11(1):24–33, January 1981.

[5] L. Moreau and C. Queinnec. Distributed and multi-type resource management. In *ECOOP'02 Workshop on Resource Management for Safe Languages*, Spain, 2002.