Self-Adapting Resource Bounded Distributed Computations

Nadeem Jamali and Xinghui Zhao
Department of Computer Science, University of Saskatchewan
176 Thorvaldson Bldg., 110 Science Place, Saskatoon, SK, Canada, S7N 5C9
n.jamali | x.zhao@agents.usask.ca

Abstract

Self-adaptation is about computations adapting to their environments. The need for adaptation may dynamically arise as a result of evolving computations or the environment. An important part of the environment is the computational resources for which computations compete.

The CyberOrgs model[1] *encapsulates distributed concurrent computations along with the computational and communication resources they require plus purchasing power for acquiring additional resources. Ownership of resources coupled with an effective control mechanism creates a predictable resource environment for computations to execute in – in a coordinated manner.*

CyberOrgs create three opportunities for self-adaptation: algorithms may be chosen using resource knowledge, additional resources may be purchased to adapt to evolving needs, and computations may coordinate use of known computational and network resources for optimal results.

The CyberOrgs model is presented and a prototype implementation is described. Our experience with using CyberOrgs' resource awareness for hierarchical coordination of distributed processor resource delivery is presented. Experimental results show that resource knowledge based reasoning leads to efficient distributed adaptation.

1. Introduction

Self-adaptation of a computation is about the computation reacting to disparities between the environment it executes in and the environment it requires for optimal execution. These disparities may result from the computation's own dynamically emerging requirements or from changes in its environment. A computation's self-adaptation may involve either altering its requirements or somehow interacting with the environment to secure what it requires.

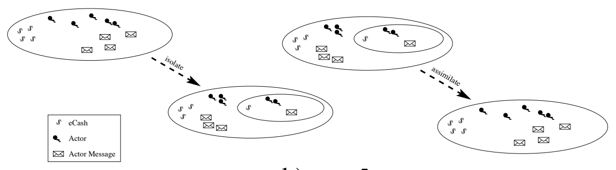
1 The model is *CyberOrgs*; encapsulations are *Cyberorgs*.

An important aspect of an environment is the resources it affords a computation. In an open system [4], the availability of resources to a computation may be threatened by other computations competing for the bounded resources. In the context of multi-agent systems, it has been shown [2] that when there are both *logical* and *resource* dependencies between agents, resource dependencies sometimes lead to logical dependencies. Coordinating resource access by agents is hence critical to reducing uncertainty and enabling agents to make control decisions – i.e., adapt – for the best global performance [8]. Furthermore, it has been argued that computation and coordination are separate and orthogonal dimensions of all useful computing [3], necessitating coordination to be addressed explicitly.

Bounded rationality limits a computation's ability to assess the situation it may need to adapt to. However, if time-bounded resource enclaves can be created for computations to execute in, they can reason about and adapt to the enclaves' boundaries. Additionally, if there is a way for computations to negotiate the boundaries of these enclaves, a richer variety of self-adaptation becomes possible.

Our approach is to create resource encapsulation for computations to execute in. We use the CyberOrgs model, which makes explicit the relationship between a concurrent computation and the resources it requires, as well as the competition between computations for bounded resources. Cyberorgs are resource encapsulations, which support computations using the resources they own. Resources are owned by cyberorgs in time and space, and can be sold to other cyberorgs for *eCash*.

This approach offers three opportunities for self-adaptation. First, with the knowledge of owned resources within the encapsulation, given the choice, a cyberorg may choose which algorithms to execute or adjust the quality of solution sought (as in Design-to-Criteria scheduling [9]). Second, the cyberorg may purchase additional resources from other cyberorgs if required. Third, the cyberorg may strategize for optimal use of owned resources.



a) `isolate` b) `asmlt`

Figure 1. Creation and Absorption

2. CyberOrgs

CyberOrgs [5] is a model for resource sharing in a network of self-interested peers, where application agents may migrate in order to avail themselves of peer-owned resources. CyberOrgs organize computational and communication resources as a market, and their control as a hierarchy. Specifically, each cyberorg encapsulates one or more multi-agent distributed computations, and an amount of *eCash* in a shared currency. Cyberorgs act as principals in a market of distributed resources, where they may use their *eCash* to buy or sell resources among themselves. A cyberorg may use the resources so acquired for carrying out its computations, or it may sell them to other cyberorgs.

CyberOrgs treat computational and communication resources as being defined in time and space. In other words, a resource is not available for use at times and locations other than when and where it exists. Sale of a resource is represented by a *contract* stipulating availability of resources to the buyer for a cost. After signing a contract, a cyberorg must migrate to the prospective host cyberorg in order to avail itself of newly acquired resources. Delivery of resources to cyberorgs is determined by a hierarchy of control decisions.

Our approach in formalizing CyberOrgs [6] is to separate computational concerns from resource concerns. We represent resource requirements of each computation by a sequence of resource ticks required to complete the computation. To simplify the model, we assume that resource requirements are known in advance. As an instantiation, we assume that the computations are carried out by systems of primitive agents called *actors* [1].

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, or use it for progressing on its application or system 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 actors, 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. To migrate a cyberorg searchs for potential hosts, attempts contract negotiation with them, and if successful, migrates.

3. Implementation of CyberOrgs

We have implemented CyberOrgs by extending *Actor Architecture* [7] – a Java library and run-time system for supporting agents. In this implementation, every agent requires processor time resource to carry out its computation, and the resource is received by the agent from the cyberorg containing it.

This implementation adds two key components to an AA platform: CyberOrg Manager and Scheduler Manager. CyberOrg Manager adds run-time support for a system of cyberorgs. A cyberorg has its own strategy for distributing available resources among its agents and hosted cyberorgs (according to relevant contracts). This strategy is encoded in a *facilitator* agent, which is a special agent that serves as the active part of a cyberorg. Among other tasks, a facilitator triggers primitive CyberOrgs operations which react to changes in the environment as well as its cyberorg's requirements.

Scheduler Manager has two parts: cyberorg scheduler and thread scheduler. The cyberorg scheduler keeps track of the CyberOrgs' hierarchical structure for a single platform and converts the hierarchical schedule represented by the structure into a flat schedule for agent threads. This schedule, which contains time intervals for which each agent is to be scheduled, is dynamically updated with changes in the hierarchical schedule resulting from invocation of CyberOrgs primitives and changes in the cyberorgs' local resource distribution. The thread scheduler schedules agent threads contained in a queue for the amounts of processor time they are supposed to be scheduled.

Distributing cyberorgs over multiple physical nodes presents implementation challenges. An internally distributed cyberorg has agents located on multiple nodes, representing its distributed parts. Each part has its own local facilitator which is responsible for making local decisions and receiving requests for primitive operations involving local agents. We use a simple protocol. The *master* facilitator maintains global information of the cyberorg, and it alone is responsible for enforcing global decisions of the cyberorg by coordinating its actions with those of other *(slave)* facilitators. By default, the master facilitator is the facilitator located at the node on which the cyberorg's creation is originally requested. Slave facilitators, by themselves, only possess the resource knowledge of their own parts of the cyberorg, and a slave facilitator can autonomously make local decisions involving agents in its own part of the cyberorg.

The way cyberorgs encapsulate computational and communication resources creates unique opportunities for scalable distributed coordination. Because delivery of network and processor resources to computations is controlled at a fine grain, idle resources are known precisely. As a result, communication and processing delays in carrying out fixed length system communications required for coordination become predictable. This – in turn – allows the distributed coordination components to reason about the feasibility of coordinated action based on good estimates of delays, and attempt only promising coordinated actions.

Coordination of distributed processor resource delivery can be treated as an example of a distributed coordination problem, illustrating how to estimate coordination delays. Consider a 2-node (N_1, N_2) request for a coordinated primitive operation to be carried out across the nodes hosting a cyberorg. The delay Δ in reaching agreement consists of several parts: D_1, the time delay from when the request is generated by the master facilitator on N_1 (or received from a slave) to when the request is scheduled to be processed on N_1; $P(a)$, the computational cost of analyzing the request and creating distributed tasks (where a is the total number of agents involved in the request); $C(N_1, N_2)$, the network delay in sending a message from N_1 to N_2 (assuming clocks are synchronized within some epsilon); D_2, the delay from when the request is received on N_2 to when the request is scheduled to be processed on N_2; $P(a_2)$, the computational cost of interpreting the request, and evaluating its feasibility on N_2 (where a_2 is the number of agents on node N_2 involved in the primitive); $C(N_2, N_1)$, the network delay of sending an acknowledgment back from N_2 to N_1; D_3, the delay from receipt of the acknowledgment to when the message is processed on N_1. Therefore, the coordination cost should be:

$$\Delta = D_1 + P(a) + (C(N_1, N_2) + D_2 + P(a_2) + C(N_2, N_1)) + D_3 \quad (1)$$

The delays D_1, D_2, and D_3 can be estimated from details of CPU scheduling. Communication costs $C(N_1, N_2)$ and $C(N_2, N_1)$ can be estimated from details of network resource control. If f_{N_i,N_j} is the network cost between nodes N_i and N_j, we can generalize equation 1 for n nodes where the master facilitator is located at N_1:

$$\Delta = D_1 + P(a) + max(f_{N_1,N_i} + D_i + P(a_i) + f_{N_i,N_1}) \quad (2)$$

for i between 2 and n. Although estimating $P(a)$ for a general purpose computation would be difficult, because we are dealing with special purpose computations for assessing feasibility of local actions, it is possible to obtain good estimates, so long as local resource availability is known, which it is in this case.

A global decision of a master facilitator may require modifying resources available to agents spread across multiple nodes. In order to guarantee that the corresponding actions associated with an n-node global decision will be performed successfully by time t on all involved nodes, a master facilitator must generate the decision by time t', so that $t' < t - \Delta$. However, some savings can be obtained by eliminating some communication. Particularly, if the master facilitator can calculate Δ without explicitly communicating with the slave facilitators, it can send requests to the slave facilitators to carry out their parts of the global action, with the knowledge that all actions will indeed succeed. Although this is not possible in general, if the the master facilitator receives periodic updates from the slaves about their locally available resources, along with promises to maintain those availabilities for certain time intervals, the master may be able to assess feasibility of remote actions so long as the actions can be completed before expiration of the resource availability promises received from the slaves.

If the coordinated action itself is required in the future, the master facilitator may estimate the delay required for agreement on feasibility of the coordinated action in a similar manner. In this case, instead of waiting for each slave facilitator to acknowledge agreement, the master facilitator may be optimistic. In other words, the slave facilitators no longer have to send acknowledgments; they only report back if they find the action infeasible. The master facilitator, in turn, waits for Δ time for possible infeasibility reports, rather than wait for each slave to acknowledge. The master would be able to calculate this Δ if the promises of resource availability received from the slaves do not expire before the slaves finish assessing local feasibilities. Additionally, instead of waiting to be informed by the master of global agreement, the slaves too optimistically wait long enough to give the master a chance to inform them about possible cancellation of the coordinated action. Because the master facilitator calculates Δ prior to communication with the slaves, it can advise the slaves in the initial communication to wait for a period $\Delta + T_r$, where T_r is the time the master would take to report cancellation to them after it has received an infeasibility report from some slave. As a result, in the case when global agreement is achieved, all parties are ready for coordinated action after a delay of $\Delta + T_r$, without any need for communication after the initial requests from the master facilitator. Furthermore, any cancellations too are known by all parties by $\Delta + T_r$.

4. Experimental Results

A number of experiments were carried out to assess the effectiveness and scalability of our approach. We collected results on delays in completing distributed schedule update tasks involving up to 1500 agents distributed over networks of two and three processors. Because a particular task would involve only a fraction of the total number of agents, this represents systems with 10^4 or more total number of agents. Specifically, we compared the delay in achieving group agreement on feasibility of success or failure of global updates to distributed processor schedules, when using and not using our approach of exploiting predictability of resource availability in cyberorgs.

We applied the approach to an implementation of CyberOrgs. In the first set of experiments – in the absence of resource availability information – we used the pessimistic approach of requiring a series of acknowledgments confirming that the requested updates can indeed be carried out at the required time. In the second set, we relied on knowledge of available resources to (optimistically) assume that the requests have been satisfied unless a failure message is received by a deadline. Note that this is not a fair comparison because in the resource unaware case, there is no guarantee of success of coordinated action until after the distributed actions have actually been attempted; nor is there a determination of failure, in which case a backtrack is required wherever the actions did happen to succeed. However, short of indicating that no comparison is possible, this appears to be a reasonable compromise.

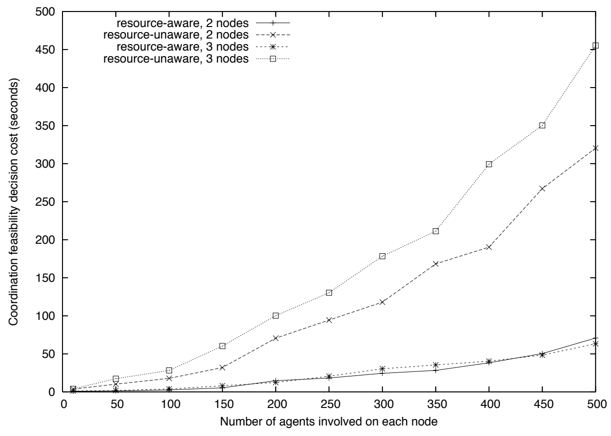


Figure 2. Delay in achieving agreement on coordinated distributed schedule update.

Figure 2 compares the delays described above for the two approaches for coordinated update of schedules for up to 1500 agents distributed across two and three physical nodes. Significant savings in the delay for global agreement on coordinated action are achieved using the CyberOrgs based resource-aware approach. These savings are in addition to the savings achieved by avoiding attempting infeasible actions, which cannot be avoided in the resource-unaware approach, even when local resource information is available. Furthermore, the penalty of increasing the number of agents linearly with the number of nodes is insignificant.

5. Conclusion

Self-adaptation is about computations staying in sync with their environments. It is difficult to reason about adapting to available resources without encapsulating computations in resource boundaries.

We described our experience with using the CyberOrgs model to address this challenge. CyberOrgs create three opportunities for self-adaptation. First, computations may use the knowledge of owned resources to choose the most suitable algorithms; second, computations may purchase additional resources to adapt to their evolving needs; third, computations may coordinate their use of known computational and network resources for optimal results.

We presented a prototype implementation which uses CyberOrgs' resource awareness for hierarchical coordination of distributed processor resource delivery. Experimental results illustrate that CyberOrgs based reasoning involving knowledge of resources aids distributed adaptation.

Although preliminary results are promising, case studies are required to obtain more authoritative results.

Acknowledgments Research funded by NSERC, Canadian Foundation for Innovation, and Saskatchewan Govt. grants.

References

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

[2] L. Gasser. DAI approaches to coordination. In N. M. Avouris and L. Gasser, editors, *Distributed Artificial Intelligence: Theory and Praxis*, pages 31–51. Kluwer Academic, 1992.

[3] D. Gelernter and N. Carriero. Coordination languages and their significance. *CACM*, 35(2):97–107, Feb 1992.

[4] C. Hewitt and P. de Jong. Open systems. In J. Mylopoulos, J. W. Schmidt, and M. L. Brodie, editors, *On Conceptual Modeling*, chapter 6, pages 147–164. Springer, 1984.

[5] N. Jamali. *CyberOrgs: A Model for Resource Bounded Complex Agents*. PhD thesis, Univ. Illinois U-C, 2004.

[6] N. Jamali and X. Zhao. Scalable hierarchical coordination of multi-agent resource usage. In *Proc. of Intl. Workshop on Massively Multi-Agent Systems*, Kyoto, Dec 2004.

[7] M. Jang and G. Agha. On efficient communication and service agent discovery in multi-agent systems. In *Proc. of SELMAS '04*, pages 27–33, Edinburgh, May 2004.

[8] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[9] T. Wagner, A. Garvey, and V. Lesser. Criteria directed task scheduling. *Journal for Approximate Reasoning–Special Scheduling Issue*, page 91.