On Cooperation Versus Competition Between Autonomous Resource Management Agents

Siri Fagernes[1]
Alva L. Couch[2]
[1]Faculty of Engineering
Oslo and Akershus University College of Applied Sciences
Oslo, Norway
[2]Computer Science Department
Tufts University
Medford, MA, USA

Abstract

A common issue in distributed systems is how to optimize resource sharing when several resource management agents have access to the same resource pool. When the total resource demand reaches max capacity of the common pool, some strategy for resource sharing must be used. We compare "altruistic" behavior in which agents give up resources according to available evidence with "selfish" approaches in which agents with priority "steal" resources from others. Through simulation, we find that "altruistic" approaches provide closer to optimum behavior than "selfish" approaches, which instead lead to instability.

1 Introduction

Traditional methods for autonomic resource management in e.g. cloud computing requires a lot of coordination and information exchange. We have previously presented a simpler approach to management ([1], [2] and [3]) based on a model of closure operators [4], [5]. Our previous results indicate that effective coordination of several resource agents is possible, even without extensive measurements and information exchange. It also appears that timing of system events affects the precision of coordination.

In this work, we study the problem of coordinating autonomous resource management agents in a setting where they are using of the same (limited) pool of resources. We are particularly interested in what type of resource sharing strategies are most effective in the situations where the resource demands exceed the total amount of available resources.

2 Related work

The traditional approach to achieving autonomic management is based on control theory. It is based on control loops that monitor and give feedback to the managed system, in

This paper was presented at the NIK-2014 conference; see http://www.nik.no/.

addition to making changes to the system based on the feedback. The control-theoretic approach is suited for managing closed systems, which are usually less vulnerable to unpredictable events and external forces influencing the system. It is not as successful representing open systems, where we do not necessarily know the inner structure and relationships [6].

The control-theoretical approach involves the use of one or more autonomic *controllers*, which sense and gather information from the environment where they reside. If any global knowledge needs to be shared among the controllers, this is normally done through a *knowledge plane* (KP) [7], [8], [9]. A KP should provide the system with knowledge about its goals and current states, and hence be responsible for gathering all necessary information and generating new knowledge and responses. This approach involves much coordination and information exchange overhead among the networked entities in the system being monitored.

To achieve autonomic resource management based upon the above approaches, one normally uses *adaptive middleware*, software that mediates between the application and the infrastructure [10], [11], [12]. This middleware mediates between managed services and clients, and reconfigures services as needed to adapt to changing needs and contingencies.

To achieve dynamic resource allocation in cloud computing, there has been recent attention to the so-called *elasticity* of cloud data centres [13], [14]. Cloud elasticity is defined as the ability of the infrastructure to rapidly change the amount of resources allocated to a service to meet the varying demands on the service while enforcing SLAs [15]. The goal is to ensure the fulfilment of SLAs with a minimum amount of overprovisioning. A common approach is to build controllers based on predictions of future load [15]. [13] proposes a system that integrates cost-awareness and elasticity mechanisms such as replication and migration. The system optimizes cost versus resource demand using integer linear programming. [15] models a cloud service using queueing theory and proposes a closed system consisting of two adaptive proactive controllers to control the QoS of a service. Predictions of future load are used as a basis for estimating the optimal resource provisioning.

In this paper, we study an approach to elasticity based upon autonomous, distributed agents. This differs from the middleware approach in that the agents are autonomous and distributed, and do not mediate between clients and services; agents simply observe what is happening and adapt accordingly. We avoid the use of a centralized planner, to increase both potential scalability and robustness, and seek instead to define autonomous, independent agents whose minimal interactions accomplish management.

3 The Resource Closure Model

The work presented in this paper is based on the work presented in [1], [2] and [3], which again is based on the work presented in [4] and [5].

The original closure model [4] consists of a single closure operator Q controlling a resource variable R. In this scenario, R represents a number of virtual servers, which has an associated cost C. C increases as R increases, and we assume a linear relationship

$$C = \alpha R, \tag{1}$$

where α is constant. This cost estimate is based upon the fact that more resources consume more power and increase wear in a roughly linear fashion.

The resource level determines the performance P of the system, which is determined based on the response time of the service that the system delivers. The response time is affected by the system load L, which is defined as an arrival rate of requests. The performance P is hence defined as the *request completion rate*. P has a baseline performance B (the quiescent request completion rate, or the performance when there is no load affecting the system), where B is a constant value. For the purposes of our study, P is defined as the baseline performance minus corrections for load and resource usage, such that

$$P = B - \frac{L}{R}. \tag{2}$$

This expression illustrates how performance increases as the resource usage increases, and decreases as system load decreases. This is a rough first-approximation of how a system would behave under load; more requests mean slower response time, but increased servers divide the load equally. Although inaccurate in practice, this function has the same rough shape and derivatives as a practical performance curve.

Decisions on resource adjustments (increase/decrease) are made based on iterative feedback on the perceived *value* V of the service. V is defined as βP, i.e. higher performance gives higher perceived value of the service. Based upon this, we obtain a total *net value* $N = V - C$, where N represents some monetary value.

Initial studies [4], [5] showed that a simple management scheme with minimal available information could achieve close-to-optimal performance.

Two closure operators

In earlier studies ([1], [2] and [3]) we extended the closure model above to apply to the scenario of two closure operators, each controlling a separate resource variable influencing the same system. In this study, we look at two variations of the two-operator model.

1. In the *independent model*, each part of the system has separate valuequatione functions V_1 and V_2, defined by

   $$V_1 = P_1 = B_1 - \frac{L}{R_1}, \tag{3}$$

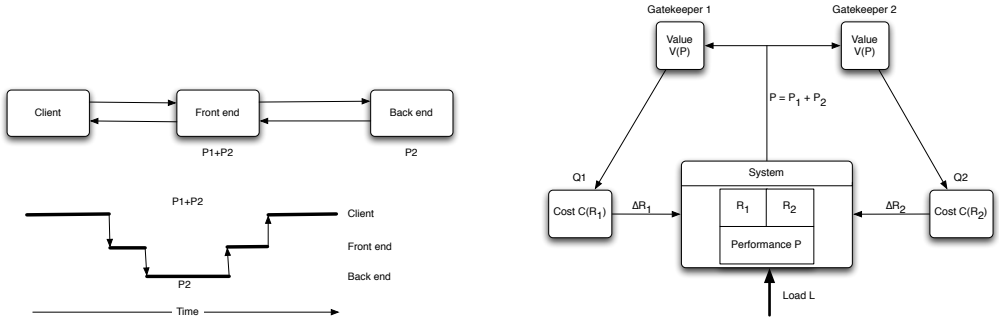
   and

   $$V_2 = P_2 = B_2 - \frac{2L}{R_2}. \tag{4}$$

   $P1$ and $P2$ represent independent performance functions. B_1 and B_2 represent the *baseline* performance or the performance when there is no load affecting the system. The factor of 2 in V_2 represents a difference in sensitivity to load.

2. The *dependent model*, where the system delivers a service with an overall performance $P = P_1 + P_2$, where P_1 (P_2) is the individual performance of the part of the system controlled by closure Q_1 (Q_2). In this case the overall value is $V = P = P_1 + P_2 = B_1 + B_2 - \frac{L}{R_1} - \frac{2L}{R_2}$.

These models are significantly different, because each closure makes decisions based on received feedback about the system response time. In the dependent model, both closures receive the same feedback, which means that they are less able to identify the effects of their own actions. In the independent model, they receive feedback of their own response



(a) Architecture of a front-end/back-end system

(b) The closure model

Figure 1: The two-closure model

times, which makes this model similar to the single-closure model. Both models are subject to resource constraints $R_1 + R_2 \leq R_{max}$. In both models, we have implemented strategies for resource sharing when the common resource pool contains less resources than the total demand from Q_1 and Q_2.

An architecture representing the two-operator scenario is illustrated in Figure 1a and the corresponding closure model in Figure 1b. Figure 1b shows the dependent model; in the independent model the system will return separate performance values P_1 and P_2 corresponding with the individual components of the system that involves the resource variables R_1 and R_2. Also, the gatekeepers in the independent model will give individual performance feedback to each of the closures controlling R_1 and R_2.

Optimal behavior

In a realistic system, the actual optimum performance of this algorithm cannot be determined; the optimum behavior depends upon a number of unobservable factors including the actual relationship between load, resources, and behavior. Our simplified model has the same character as the realistic one, with one important exception: the optimum behavior is known and can be compared with system response.

There are two cases. If the system is unsaturated, in the sense that enough resources are available to satisfy demand, the theoretical optimum can be found by setting the derivative to zero. This gives the following theoretical optimal resource levels in the unsaturated case:

- $R_1^O = \sqrt{L}$
- $R_2^O = \sqrt{2L}$

If the system is saturated, in the sense that the theoretical best values are higher than the resource pool allows, we must compute separate optimum values. The system is at saturation when

$$R_1 + R_2 = R_M \tag{5}$$

and $\frac{dN}{dR_1}, \frac{dN}{dR_1} > 0$. At saturation, the theoretical optimum must be along the line $R_1 + R_2 = R_M$. By substituting this into the equation for N, we recompute the derivative of N and obtain the theoretical optima for R_1 and R_2 in the case of saturation:

$$R_1^O = R_M(\sqrt{(2)} - 1), \tag{6}$$

and

$$R_2^O = R_M(2 - \sqrt{(2)}). \tag{7}$$

These are *theoretical* optimal values, which are not known to the system being simulated. These are only used to evaluate the performance of the system. We have contrived to know the theoretical optima via our choice of models.

Resource sharing strategies

When resource usage is unlimited, resource allocation happens through each closure operator making a decision to either increase or decrease the current resource level. The size of the increment/decrement unit is determined by the fixed increment size (which is set to 3 resource units in the simulations).

When the resource pool is shared and of limited size, one may experience that the total sum of resource allocation needs exceed the capacity of the pool. This means that the operators will not be able to achieve their optimal resource usage level. To achieve the best overall system performance, given the resource constraints, priority should ideally be given to the operator that can gain the best performance gain from increasing (or avoiding to decrease) its' current resource usage level. This could mean, for instance, that one operator would have to give up resources to make them available to another operator with a more pressing need.

In our closure models, the resource controllers are *autonomous*, hence there are no priority mechanism enforced on the operators. If any of the operators can gain more than the other from increasing their resource level, any renunciation of resources from the other operator will be the result of a voluntary action.

For the competition scenario, we have implemented two methods for resource allocation/sharing, which are chosen in the absence of a priority mechanism.

1. *First-come, first-served:* The simplest strategy, which says that if a closure operator asks for more resources than what is available in the pool, it receives what is left. If the pool is empty, nothing is given. The pool will then not have anything available until the agents decrease their resource level.

2. *Voluntary handoff:* This is an altruistic strategy. The agents exchange information about their current dV/dR-values, and if the resource demand exceeds the available resources the agent with the lowest resource pool will give priority to the other agent. We have tested three different versions of the voluntary handoff-strategy that differ in the decrements made to their own resources based upon perceived need of others.

   (a) *No increment*: An agent which originally has made the decision to increase the resource level, remain at the same level if the other agent has a higher dV/dR-value.

   (b) *Decrement of 1 unit*: The agent with the lowest dV/dR, reduces its current resource level with 1 decrement unit (which in the simulations is 3 resource units)

   (c) *Decrement of 2 units*: The agent with the lowest dV/dR, reduces its current resource level two decrement units (which in the simulations is 6 resource units).

4 Experiments

In this section the experiment setup will be briefly explained. We have run simulations on the independent and the dependent model, under the same conditions. System load L is sinusoidal, $L = 1000sin(t/p) * 2\pi + 2000$, which makes the load vary periodically between 1000 and 3000. The models are event-based, and we have tested both generating *synchronous* events and probability-based or *asynchronous* events.

- *Synchronous behavior:* the resource controllers would adjust their values at exactly the same time.
- *Asynchronous behavior:* everything that happens in the system; resource updates, system response measurements, and load updates were treated as probability-based events in time. This enabled us to model inaccuracy in the available information due to delays in updates and measurements, and how this affected whether optimal resource usage can be achieved.

To check how resource constraints affected the system, we varied the size of the common resource pool. For the rounds when the resource pool was smaller than the total resource demand we compared the strategies *first-come, first-served* and *voluntary handoff* with back off, or return of 1 or 2 units.

5 Results

In this section the main findings are presented.

Two independent agents operating in the same environment

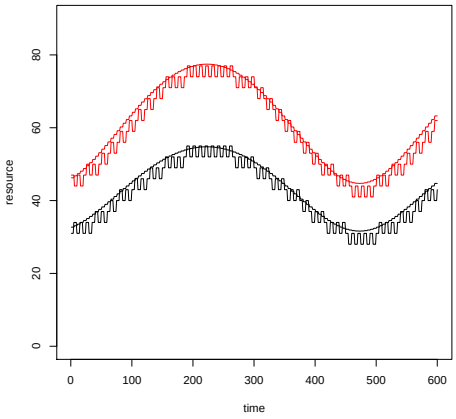
Two independent agents operating in the same environment converge to and track their individual optima. In earlier studies, a single agent with access to "full information" (knowledge of current system load, resource usage, and value estimates) converged to its theoretical optimum. As shown in Figure 2, this result still holds when we simulate two agents in the same environment.

Behavior with limited resources

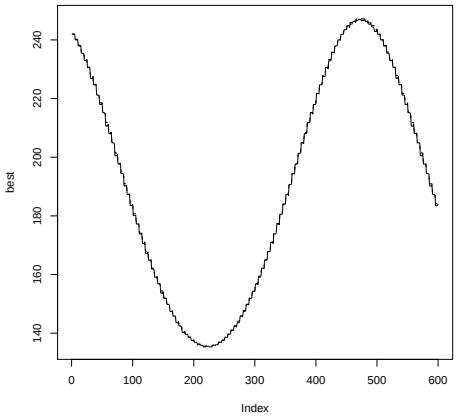
The system tracks the optimum even if resources are limited. The two-operator closure model (independent and dependent version) converge to the theoretical optimum when there are limitations on resource usage that lead to competition between the agents. If the resource constraints do not allow the system to reach its unbounded optimum, the system will converge to the bounded optimum (Figure 3). In the figures presenting simulation results under saturation, two different theoretical optima are displayed. The straight horizontal lines represent the theoretical optima under saturation for each of the operators (Equation 6 and 7), while the dotted sine curves represent the unbounded theoretical optima.

Saturation is partly addressed by altruism

In the saturated case, returning resources to the pool is the better resource sharing strategy. All simulations in this study (independent vs dependent model, synchronous vs asynchronous variable updates) show that the better solution to resource sharing is that the agent with the lowest current dV/dR-value decreases its resource usage (and hence

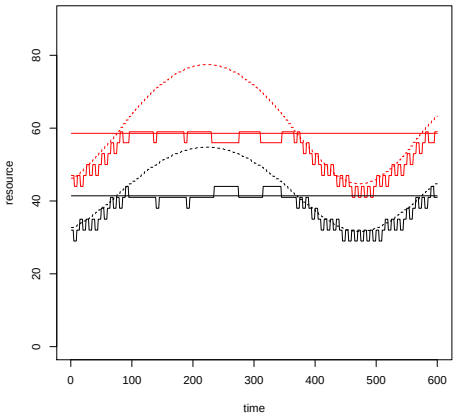


(a) Resource usage

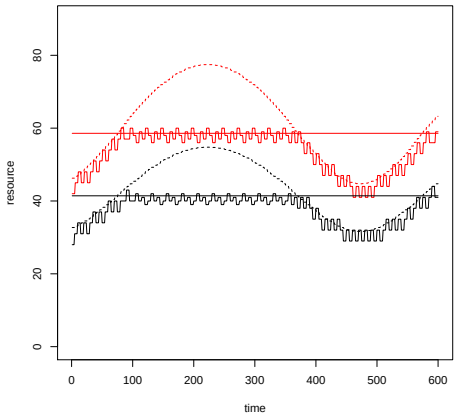


(b) Net value

Figure 2: Two operators (independent model). No saturation ($R_M = 200$). Two independent operators tracks their optimum with very low error. Achieved net value very close to the theoretical best value.



(a) "first-come, first-served".



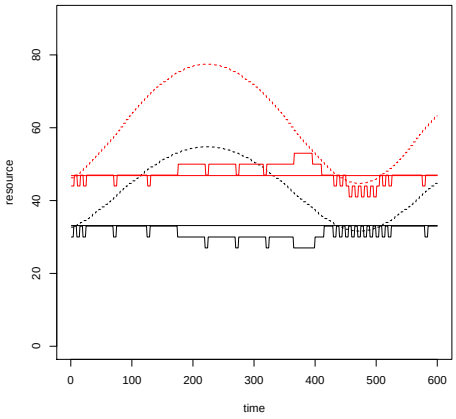
(b) Voluntary handoff (decrement -2)

Figure 3: Two operators (independent model) under saturation ($R_M = 100$).

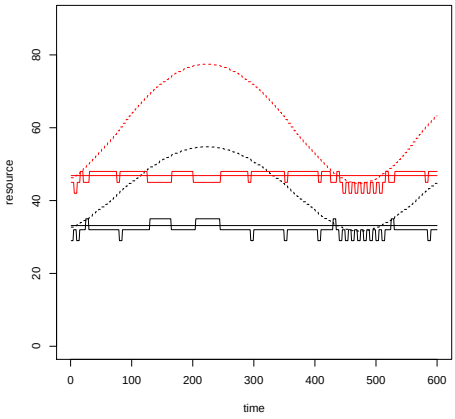
increases the size of the common resource pool). The cost of using this strategy is that is requires more information exchange among the resource controllers, to decide which one should have priority when sharing the resources.

The strategy of *first-come, first-served* is the least successful of those tested. Which agent receives more resources is arbitrary compared to when using the altruistic strategy; it simply depends on the order of requests to the pool. While the "winning" resource controller achieves higher individual value, the overall system achieves less optimal performance (Figure 4a).

We also see that the deviation from the optimum when using this strategy increases when the resource pool is smaller relative to the overall resource demand.

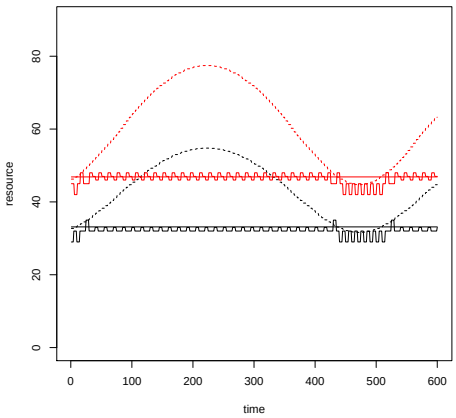


(a) "First-come, first-served".

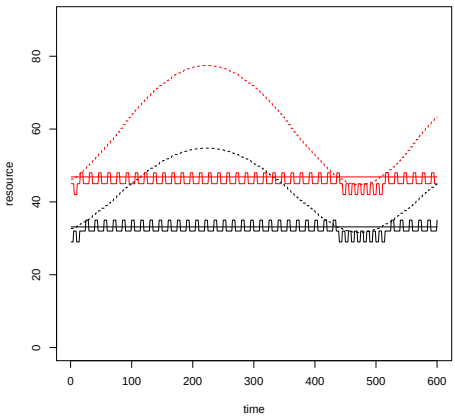


(b) Voluntary handoff (decrement 0)

Figure 4: Two operators (independent model) under saturation ($R_M = 80$).



(a) Voluntary handoff (inc=-1).



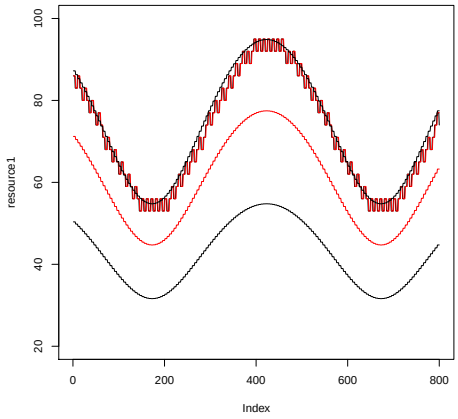
(b) Voluntary handoff (inc=-3).

Figure 5: Two operators (independent model), saturation ($R_M = 80$). Voluntary handoff with varying decrement sizes.

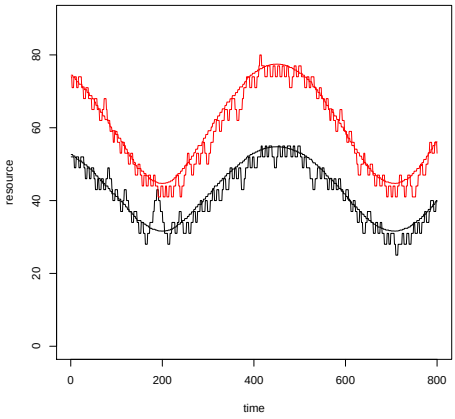
Timing of events and optimal performance

Synchronous updates gives false optima in the dependent model, while in the independent model, timing is less important. In general, synchronous updates result in a closer fit than asynchrony when avoiding false optima.

As we have seen in previous studies of the dependent model with identical value functions, adjusting both resource variables synchronously can lead to convergence to a so-called *false optimum* (ostensibly because the individual agents cannot distinguish between their actions and the actions of others). Figure 6a illustrates that this also happens for the scenarios where the value functions are different.The oscillating curves (two on top of each other) represents the actual resource usage, and the upper solid sine curve represents the computed false optimum. The two lower sine curves represents the theoretical optima for R_1 and R_2. Both closure operators converge to a resource level higher than their individual theoretical optima. This shows that synchronous updates

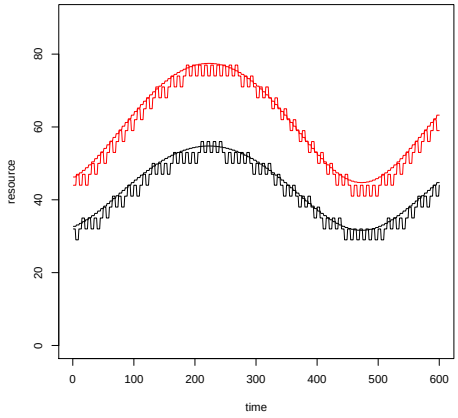


(a) Synchronous updates. The upper sine curve represents the false optimum.

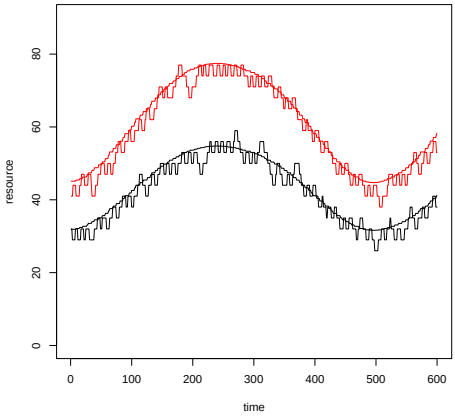


(b) Asynchronous updates.

Figure 6: Two operators (dependent model). $R_M = 200$, no saturation.



(a) Synchronous updates.



(b) Asynchronous updates.

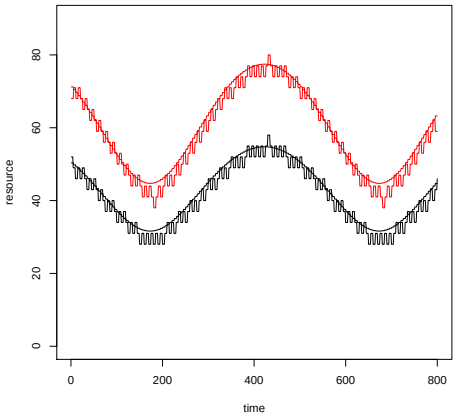
Figure 7: Asynchronous vs synchronous updates in the independent model.

of R_1 and R_2 result in false optima even when the theoretical optima for R_1 and R_2 differ. Asynchronous updates still remove the false optima when value functions are different (Figure 6b). Both closure operators converge to their theoretical optima. Thus asynchronous updates of R_1 and R_2 address the false-optima problem also when the theoretical optima for R_1 and R_2 are different.

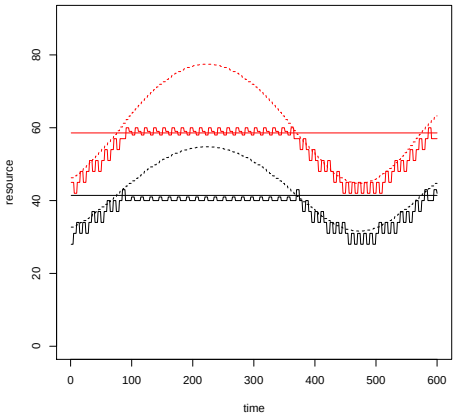
In the independent model, each resource controller receives individual feedback on performance, which may explain why the false optimum-problem does not occur in these simulations. Synchronous updates (Figure 7a) result in a closer convergence to the optimum compared with asynchronous adjustments (Figure 7b).

Removing errors via constraints

Errors due to timing problems can be removed by adding resource constraints. In previous studies we have observed how full synchrony in the resource updates can result in



(a) Resource constraints ($R_M = 150$), but no saturation.



(b) Saturation, $R_M = 100$.

Figure 8: Two operators (dependent model), synchronous updates. This shows that resource constraints remove false optima (as seen in Figure 6a).

convergence to the wrong resource level ("false optimum"). An example of this can be seen in Figure 6a. Adding resource constraints removes this phenomenon, and makes the synchronous updates the most efficient strategy (Figure 8a and 8b).

In the dependent model (where the controllers receive feedback based on the overall performance, not their individual performance), the phenomenon we have referred to as *false optima* frequently occurs when the controllers adjust resource values in full synchrony.

When both resource variables are starting at the same initial values, and are updated synchronously, we get the following situation. Each of the controllers see the total result, and hence believes that the change in system value is based solely on their own resource adjustments. The estimated (false) N would then be

$$N = B - \frac{L}{R} - \frac{2L}{R} - R \tag{8}$$

which would give the false optimal value

$$R_f^o = \sqrt{(3L)} \tag{9}$$

for both variables R_1 and R_2. In Figure 6a we see how both resource controllers hit the false optimum (top black line).

6 Conclusions

The purpose of this study has been to get a deeper understanding of the implicit interaction between two autonomous agents in a competitive environment. The two-operator closure model still converges to the theoretical optimum when the system is under bounds on resource usage. If the resource constraints do not allow the system to reach its unbounded optimum, the system will converge to the bounded optimum. If the agents differ in potential gain from resource increase, the system performs best when the agent with the lowest potential gain voluntarily reduces its resource level. Based on the findings

in this study, theoretical optimal behavior can be achieved without heavy computation or extensive information exchange. The agents need only to estimate which one would benefit most from increasing their resource usage level.

Timing of system events affects the precision of the model. Synchronous updates makes the dependent model vulnerable to settling at the wrong resource levels, while it seems to give the closest fit for the independent model. Convergence to the false optima can be avoided by adding stronger bounds on resource usage.

The next step in this work is to build larger scenarios of several agents, to be able to verify whether the results we have observed so far can be generalized to large-scale systems. So far we have not discussed implementations of the closure model, as this is also considered part of future work. However, the point of this design that it is easy to implement, because there are decoupled sensors and minimum information exchange.

References

[1] S. Fagernes and A. Couch. On the behaviour of autonomous resource management agents. In *LNCS 6155 AIMS'10*, 2010.

[2] S. Fagernes and A. Couch. On alternation and information sharing among cooperating autonomous management agents. In *Proc. of MENS 2010*, 2010.

[3] S. Fagernes and A. Couch. Coordination and information exchange among resource management agents. In *Proc. of IM2011*, 2011.

[4] A. Couch and M. Chiarini. Dynamics of resource closure operators. *Proceedings of Autonomous Infrastructure Management and Security 2009, Twente, The Netherlands, June 30-July 2, 2009*, 2009.

[5] A. Couch and M. Chiarini. Combining learned and highly-reactive management. *Proceedings of Managing Autonomomic Communications Environments (MACE) 2009, Venice, Italy, Oct 27, 2009.*, 2009.

[6] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259, 2006.

[7] D. F. Macedo, A. L. dos Santos, J. M. S. Nogueira, and G. Pujolle. A knowledge plane for autonomic context-aware wireless mobile ad hoc networks. *Management of Converged Multimedia Networks and Services (LNCS)*, 2008.

[8] M. Mbaye and F. Krief. A collaborative knowledge plane for autonomic networks. *Autonomic Communication*, 2009.

[9] D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski. A knowledge plane for the internet. *Proc. of SIGCOMM'03*, 2003.

[10] P. Padala, K. G. Shin, X. Zhu, M. Uysal, S. Singhal Z. Wang, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. *EuroSys*, 2007.

[11] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. Dynamic estimation of CPU demand of web traffic. *Valuetools'06:Proceedings of the 1st international conference on Performance evaluation methodolgies and tools.*, 2006.

[12] C. Adam and R. Stadler. Service middleware for self managing large-scale systems. *Network and Service Management*, 4(3):50–64, 2008.

[13] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. Kingfisher: Cost-aware elasticity in the cloud. In *INFOCOM, 2011 Proceedings IEEE*, pages 206–210. IEEE, 2011.

[14] Pablo Chacin and Leando Navarro. Utility driven elastic services. In *Distributed Applications and Interoperable Systems*, pages 122–135. Springer, 2011.

[15] Ahmed Ali-Eldin, Johan Tordsson, and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.