A Reinforcement Learning Framework for Dynamic Resource Allocation: First Results*

David Vengerov, Nikolai Iakovlev
Sun Microsystems Laboratories
UMPK16-160, 16 Network Circle
Menlo Park, CA 94025
david.vengerov@sun.com

Abstract

This paper addresses the problem of dynamic resource allocation among multiple entities sharing a common set of resources. A solution approach is presented based on combining the reinforcement learning methodology with function approximation architectures. An implementation of this approach in Solaris 10 demonstrated a robust near-optimal performance on a simple problem of transferring CPUs among resource partitions so as to match the stochastically changing workload in each partition, both for large and small CPU migration costs.

1. The Problem

Developing computing systems that are self-configuring and self-optimizing in unpredictable environments is becoming the central concern of industrial giants such as IBM, HP, and Sun. For example, the recently popular "utility computing paradigm describes the environment where system components (agents) "purchase" resources from each other to respond to local spikes in demand.

While this vision is very promising, only small steps have been made so far toward achieving it. Developing self-adaptive policies for dynamic resource allocation (DRA) is the key issue in developing self-optimizing utility computing systems. Sun Microsystems has enabled this capability in its Solaris 10 operating system and the DRA approach presented in this paper is evaluated in that domain.

*This material is based upon work supported by DARPA under Contract No. NBCH3039002. The authors would like to thank Declan Murphy from Sun Microsystems for helpful comments and corrections to this paper.

The general DRA framework applies to a computing facility with a pool of shared resources (CPUs, memory, bandwidth, storage space, servers, etc.). Such a facility can be a data center, a supercomputer, or a multi-processor machine running an operating system supporting dynamic reconfiguration of resources. Assume that N clients (projects) are using this facility simultaneously and that resources can be dynamically reassigned among the projects. We aim to develop a scalable algorithm for such a reassignment so as to maximize the *productivity* of the computing facility (the long term sum of utility/revenue received from using the resources) under *unknown and stochastic* workload characteristics for different projects, such as arrival rates and resource requirements.

2 Solution Methodology

Let x^i be the amount of currently allocated resources to project i and u^i be its current resource utilization. Let $U^i(s)$ for project i be the expected average utility per unit of time received in the future by that project starting from state $s = (x, u)$ under the current resource allocation policy. Once such utility functions are obtained, a large variety of resource allocation methods can be used to reallocate resources among the agents so as to maximize $\sum_i U^i - C$, where C is the total resource transfer cost.

The key problem addressed in this paper is that of learning U^i in the dynamic resource allocation setting. This problem is naturally formulated using the Reinforcement Learning (RL) framework [1]. This approach is used in [2] for the case where the state of each project can be described using a single state variable that can take only a small number of values. We demonstrate a successful application of RL to a more realistic scenario when the state variables in each project can be contin-

uous (e.g., resource utilization level u). In this case, the utility value has to be generalized across similar states, some of which have never been visited before. To achieve this, an agent is assigned to each project that uses a *parameterized function approximation architecture* $\hat{U}^i(s,p)$ to approximate $U^i(s)$. For simplicity, we use a linear combination of basis functions: $\hat{U}(s,p) = \sum_{j=1}^{M} p^j \phi^j(s)$, where $p = (p^1, p^2, ..., p^M)^T$ is a vector of tunable parameters.

The RL framework applies to a Markov Decision Problem (MDP), where after taking action a_t in state s_t at time t, the agent's state changes stochastically to s_{t+1} and some reinforcement $r(s_t, a_t, s_{t+1})$ is observed. The the following vector parameter updating rule can then be used [1]:

$$p_t = p_t + \alpha_t [r(s_t, a_t, s_{t+1}) + \gamma \hat{U}(s_{t+1}, p_t) - \hat{U}(s_t, p_t)] \phi(s_t), \quad (1)$$

where α_t is the learning rate and $\phi(s) = (\phi^1(s), \phi^2(s), ..., \phi^M(s))^T$ is a vector of all basis functions.

The parameter updates in equation (1) are guaranteed to converge in the single agent case if the policy for selecting actions in every state remains fixed and certain theoretical conditions are satisfied [3], such as linear independence of the basis functions $\phi^i(s)$. However, instead of evaluating a given policy we usually want the agent to learn the *optimal* policy. For example, in the DRA problem we want the agent to choose the action that maximizes $\sum_i \hat{U}^i(s,p) - C$, where $\hat{U}$ changes as the learning progresses. Unfortunately, the convergence guarantee no longer applies in this case. Moreover, since all agents are changing their $\hat{U}(s,p)$ and hence their behavior policies, each agent faces a non-Markovian, non-stationary learning environment. Despite these issues, our experiments show that RL can still be used to learn a near-optimal resource allocation policy.

3 Experimental Setup and Results

Due to space limitations in this short paper, we only give a brief description of the results of applying the DRA-RL approach to the problem of migrating CPUs between partitions in Solaris 10 in response to a stochastically changing workload. A fuzzy rulebase with tunable output parameters is an instance of a linear function approximation architecture with convenient knowledge representation/extraction capabilities [4], and it was used for approximating U^i for each project. The following resource allocation policy was used at regular time intervals: each agent i calculated the change in its $\hat{U}^i$ if a unit of resources were added or removed, and resources were then taken away from the least needy agent and given to the most needy one as long as the combined increase in utilities outweighed the resource transfer cost.

The reinforcement signal for each agent was: $r(\tilde{s}_t, s_{t+1}) = N_t(\bar{u}_{t+1} - C_h) - n(t)C_{tr}$, where N_t and $\bar{u}_{t+1}$ are the number of CPUs and average CPU utilization in the partition between states $\tilde{s}_t$ and s_{t+1}, C_h is a *holding cost* per CPU, C_{tr} is the *transfer cost* per CPU and $n(t)$ is the number of CPUs transferred at time t in or out of this partition. The above reinforcement function reflects the UC environment, where each job brings a fixed utility upon completion and resources can be purchased at the unit cost of C_h.

We used two benchmark policies representing extremes of this policy space: a "constant" policy of performing no CPU transfers and keeping an equal number of CPUs in each partition and an "aggressive" policy – transferring one CPU from partition i to partition j if partition j had a higher CPU utilization. As expected, the constant policy was optimal for a large transfer cost and the aggressive policy was optimal for a small transfer cost. The policy learned using the DRA-RL architecture was able to match the performance of the optimal policy in each scenario starting from parameter vector $p = 0$, demonstrating robustness of the DRA-RL framework. This framework can solve much more general problems than the one described above (e.g., transferring multiple units of heterogeneous resources), and we are currently conducting such experiments.

References

[1] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[2] G. Tesauro, "RL-Based Online Resource Allocation in Multi-Workload Computing Systems," Presentation at AAAI Fall Symposium on Real-Life Reinforcement Learning, Washington DC, Oct. 22, 2004.

[3] J. N. Tsitsiklis and B. Van Roy, "An Analysis of Temporal-Difference Learning with Function Approximation," *IEEE Transactions on Automatic Control*, Vol. 42, No. 5, May 1997, pp. 674-690.

[4] D. Vengerov, *Multi-Agent Learning and Coordination Algorithms for Distributed Dynamic Resource Allocation.* Ph.D. Dissertation, Department of Management Science and Engineering, Stanford University, March 2004. Available electronically at http://research.sun.com/people/vengerov/dissertation.pdf