Fundamentals of Decentralized Optimization in Autonomic Systems

Tomasz Nowicki, Mark S. Squillante, Chai Wah Wu
Mathematical Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA

1 Introduction

An autonomic system is a complex information system comprised of many interconnected components operating at different time scales in a largely independent fashion that manage themselves to satisfy high-level system management requirements and specifications [5]. This includes providing the self-$*$ properties of self-configuring, self-repairing, self-organizing and self-protecting. A fundamental problem for achieving the goals of self-management concerns the general optimization framework that provides the underlying foundation and supports the design, architecture and algorithms employed throughout the system. Given the increasing complexity of current and future information systems, a decentralized approach is a natural way to design and implement autonomic systems that provide self-$*$ properties. On the other hand, a centralized approach with complete knowledge over all constituent system components has the potential to provide significant improvements over a decentralized approach, in the same way that solutions to global optimization problems (if attainable) are often superior to the corresponding locally optimal solutions. This fundamental problem involving the tradeoff between centralized and decentralized approaches arises in a wide range of applications, and its solution is especially important to achieve the goals of self-management in autonomic systems.

Our study focuses on this fundamental problem within the context of the decentralized optimization aspects of self-management as a step toward providing a scientific basis for information systems that provide self-$*$ properties. Specifically, we consider a hierarchical and decentralized framework for optimal self-management in which a complex information system is partitioned into multiple application environments, AE, each of which has an application manager, AM, that controls and optimizes the resource management and operations within the AE. The collection of AMs are in turn controlled and optimized by a central manager, CM, that allocates the system resources among the AEs. The system hierarchy can contain many levels, i.e., each AE can itself include a manager controlling and optimizing subenvironments within the AE. It is the goal of the CM to optimize the overall objective function for the entire information system. This objective function can be based on any combination of measures (e.g., availability, costs, overheads, penalties, performance, reliability, revenues, risk, robustness, utility [6]) and depends upon the resources, workloads and other parameters of the information system. This objective is also based on the self-$*$ property of interest, each of which should involve optimization in an important role.

Independent of the details of the objective function, a central question concerns whether there is any loss in optimality when the self-management is carried out via a decentralized approach. One purpose of this paper therefore is to provide conditions under which a decentralized optimization framework is as good as a centralized framework. In particular, we show that there is no loss of quality in the optimal self-management of complex information systems when a decentralized approach is used and we provide a foundation for the decentralized approach to designing and implementing autonomic systems with self-$*$ properties. Another purpose of our study is to investigate in more detail the interactions between system components at different levels of this hierarchical decentralized framework for optimal self-management. Specifically, we consider a negotiation scheme where additional information is passed between the CM and the AEs in order to significantly increase the efficiency with which the optimization algorithms compute the optimal solution. We then exploit a representative example of our general mathematical framework to investigate other fundamental properties of decentralized optimal self-management in practice, including phase transitions, chaotic behavior, stability and computational complexity.

2 Optimal Total Cost

To formally express the problem in the standard form of optimization problems, we consider without any loss

of generality minimizing a *cost* function (since maximizing f is equivalent to minimizing $-f$). A cost function $f_i(x_i, r_i, u_i)$ is associated with each AE_i, where x_i is the set of variables that can be changed in AE_i, r_i is the set of resources allocated to AE_i, and u_i is the set of external variables that affect AE_i. Examples of r_i include the set of servers assigned to AE_i, or the amount of disk space or processing time made available to AE_i. Examples of u_i include the current workload of AE_i or conditions imposed by external events such as failures. The set of variables x_i must also satisfy the set of constraints C_i, *i.e.*, $x_i \in C_i$ is the feasible region of operation for AE_i. In general, C_i will depend on r_i and u_i. Examples of C_i include conditions imposed by AE_i, such as an upper limit on the percentage of requests having end-to-end response times that exceed some threshold when the resources of AE_i are organized in a multi-tier architecture.

The total cost function for the entire system is given by $h(f_1(x_1, r_1, u_1), \ldots, f_n(x_n, r_n, u_n))$, where h aggregates the cost of each AE_i into a single total cost. Examples of h of interest include summation (SUM), the maximum function (MAX), and the minimum function (MIN).

The total set of resources in the system is finite, and the set of resources assigned to the AEs is required to satisfy a constraint: $(r_1, \ldots, r_n) \in R$. Examples of this constraint include bounds on the amount of disk space or number of servers, or in the case when the servers are organized in multiple tiers, bounds on the end-to-end response or processing time. By allowing elements of R to represent a strict subset of the resources, the CM can reserve a set of available resources for direct allocation to any AE_i rather than being moved from AE_j to AE_i.

Then the goal of the autonomic system is to globally minimize the total cost function h subject to constraints:

$$h_c \;=\; \min_{x_i, r_i} h(f_1(x_1, r_1, u_1), \ldots, f_n(x_n, r_n, u_n)) \quad (1)$$

such that (s.t.) $(r_1, \ldots, r_n) \in R$, $x_i \in C_i(r_i, u_i)$. The value h_c is the optimal cost of the system (which in general depends on the set of external variables $u_1, \ldots, u_n$) in a *centralized* framework as it is the globally minimal cost among all feasible resource allocations r_i and all feasible sets of variables x_i. The cost h_c can be computed by an optimization algorithm which has knowledge about the operations of all AE_i including the cost functions f_i.

Now we seek to find conditions under which h_c can be obtained using a hierarchical, decentralized framework.

3 Decentralized Optimization

For each AE_i, the corresponding AM_i minimizes the cost function for AE_i by solving the optimization problem:

$$g_i(r_i, u_i) \;=\; \min_{x_i} f_i(x_i, r_i, u_i) \quad (2)$$

s.t. $x_i \in C_i(r_i, u_i)$, where r_i are the set of resources allocated by the CM to AE_i. In turn, the CM determines the resource allocation by solving the optimization problem:

$$h_d \;=\; \min_{r_i} h(g_1(r_1, u_1), \ldots, g_n(r_n, u_n)) \quad (3)$$

s.t. $(r_1, \ldots, r_n) \in R$.

Notice the decentralized nature of this scheme. Each AE_i optimizes the cost within its environment and passes this optimal cost to the CM. In particular, there is no need for the CM to know the form of the cost function f_i. Additional information can be sent from each AE_i to the CM to aid in the optimization of the total cost; see §4.

For vectors in $\mathbb{R}^n$, let $\geq$ be the partial order generated by the positive orthant, *i.e.*, $x \geq y$ if $x_i \geq y_i$ for all i.

Definition 1 *A function $g : \mathbb{R}^n \to \mathbb{R}^m$ is called* order-preserving *with respect to $\geq$ (OPGT) if $g(x) \geq g(y)$ whenever $x \geq y$.*

Examples of OPGT functions are SUM, MAX and MIN.

Suppose the external variables u_i are constant or slowly varying compared to the optimization so that we can ignore them for the moment; see §5 where this is addressed.

Theorem 1 *If the aggregation function h is OPGT, then $h_c = h_d$,* i.e., *decentralized optimal self-management is as good as centralized optimal self-management.*

Proof: Clearly $h_d \geq h_c$. Let x_i^* and r_i^* be the optimal set of variables and resource allocations s.t. $h(f_1(x_1^*, r_1^*, u_1), \ldots, f_n(x_n^*, r_n^*, u_n)) = h_c$ while satisfying $(r_1^*, \ldots, r_n^*) \in R$, $x_i^* \in C_i(r_i^*, u_i)$. Then by definition $g_i(r_i^*, u_i) \leq f_i(x_i^*, r_i^*, u_i)$, and from the OPGT property of h we have:

$$\begin{aligned} h_d &\leq h(g_1(r_1^*, u_1), \ldots, g_n(r_n^*, u_n)) \\ &\leq h(f_1(x_1^*, r_1^*, u_1), \ldots, f_n(x_n^*, r_n^*, u_n)) = h_c. \end{aligned}$$

4 Hierarchical Negotiation

In general, continuous optimization algorithms to solve Eq. (3) perform much better if in addition to the ability to evaluate the objective function $\tilde{h}(r_1, \ldots, r_n) = h(g_1(r_1, u_1), \ldots, g_n(r_n, u_n))$, the gradient $\nabla\tilde{h}$ of the objective function is also available.

Note that $\nabla\tilde{h} = \sum_i \nabla_i h \cdot \frac{\partial g_i}{\partial r}$ with $\frac{\partial g_i}{\partial r_j} = 0$ for $i \neq j$. Assuming the constraints $x_i \in C_i(r_i, u_i)$ are written as $c_i(x_i, u_i) = r_i$ or as $c_i(x_i, u_i) \leq r_i$, then $-\frac{\partial g_i}{\partial r_i}$ are the Lagrange multipliers ($\mathcal{LM}$s) in solving Eq. (2); refer to [1]. Thus by having each AM_i send to the CM both $g(r_i, u_i)$ and the corresponding $\mathcal{LM}$s, the gradient $\nabla\tilde{h}$ can be efficiently computed by the CM. In this case, the (logical) negotiation scheme between the CM and the AEs is as follows:

1. The CM sends r_i to each AE_i.

2. Depending on the architecture of the system, the CM might also send the set of external variables u_i to each AE_i. In other cases, the external variables u_i are readily available to each AE_i.

3. Each AE_i computes $g_i(r_i, u_i)$ and sends it to the CM along with the corresponding $\mathcal{LM}$s.

4. The CM uses this information to compute $\tilde{h}$ and $\nabla\tilde{h}$ and find the next resource allocation $(r_1, \ldots, r_n)$.

5. This is iterated until a suitable resource allocation is found or the algorithm converges.

In the beginning, each AE_i does not have to compute $g_i(r_i, u_i)$ too accurately, e.g., run too many iterations within AE_i to compute $g_i(r_i, u_i)$.

For environments where derivatives are not available or cannot be computed efficiently, the above negotiation approach can be used together with derivative-free optimization (DFO) methods [2] to realize similar implementation benefits. In particular, when DFO is used to compute g_i, the trust region radius and (internal) trust region model used in computing g_i can be sent to the CM instead of the $\mathcal{LM}$s. This information can be sent to the CM in a compact form and used in an efficient manner that is analogous to the five-step negotiation scheme provided above.

5 Example

Let us now consider a representative example in which a set of M heterogeneous computing servers, $S_1, \ldots, S_M$, and a set of routers are used by a common service provider to host a set of N client environments, $AE_1, \ldots, AE_N$. The router assigned to AE_i immediately routes all incoming requests to one of the servers allocated to and under the control of AM_i. A service-level agreement (SLA) is created for each AE to define the corresponding quality-of-service (QoS) requirements and the revenues (respectively, penalties) for satisfying (respectively, failing to satisfy) these requirements. To elucidate the exposition, we consider SLAs with a single QoS class within each AE.

Self-optimization in such an autonomic system includes the allocation of servers among the set of AEs, the routing of requests within each AE, and the scheduling of requests at each server within an AE, all in order to minimize the global objective function based on the collection of SLAs. Specifically, each AM_i solves the optimization problem in Eq. (2) and the CM solves the optimization problem in Eq. (3), where r_i are the set of servers and the router allocated by the CM to AE_i, $(r_1, \ldots, r_N) \in R$, u_i are the set of workload characteristics for AE_i, and $x_i \in C_i(r_i, u_i)$ are the set of router and per-server scheduling variables that can be changed in AE_i. With our focus here on single-class QoS requirements, there is no need to set or adjust any scheduling variables at each server, and thus x_i consists solely of a vector of the proportional weights for routing requests among the set of servers allocated to AE_i. The set R is the set of all possible N-way partitions of the set of servers $\{S_1, \ldots, S_M\}$, together with a router for each AE.

The workloads u_i can be accurately modeled as stochastic processes that vary over time [3]. To achieve the global objectives under such nonstationary behavior, the resource allocation decisions are made periodically at time epochs t_ℓ, $\ell = 0, 1, \ldots$. The time scales at which these scheduling decisions are made depend upon several factors, including the delays, overheads and constraints involved in making changes to decision variables, the QoS requirements of each AE_i, and the properties of the underlying stochastic processes. Then the optimization problems in (2) and (3) are solved at each scheduling epoch t_ℓ based on measurements collected during previous scheduling intervals $\tau_k \equiv [t_{k+1}, t_k)$, $k = 0, \ldots, \ell-1$, in order to determine the optimal variables x_i^* and r_i^* that should be deployed during the next scheduling interval τ_ℓ.

The details of the cost functions in Eqs. (2) and (3), as well as the corresponding constraints $C_i(r_i, u_i)$, depend upon the specific client environments being served. We shall thus focus on a typical scenario in which the QoS requirements are based on the response times of client requests. In particular, $f_i(x_i, r_i, u_i) = f_i(\mathsf{E}[T_i(x_i, r_i, u_i)])$ where $\mathsf{E}[T_i(x_i, r_i, u_i)]$ denotes the expectation of the stochastic response time process for AE_i given the allocation of resources r_i and under the routing and scheduling variables x_i and the workload u_i. The aggregate cost function is given by $h(g_1(r_1, u_1), \ldots, g_N(r_N, u_N)) = \sum_{i=1}^{N} g_i(r_i, u_i)$, although MAX and MIN could be used instead of SUM in a similar fashion.

Now we can consider each AE_i during any scheduling interval τ_ℓ in which the corresponding workload processes u_i are stationary. Every AM_i determines the optimal routing variables $x_i^* \in C_i$ by solving the optimization problem with $f_i(\mathsf{E}[T_i(x_i, r_i, u_i)])$ substituted in Eq. (2), and the CM determines the optimal allocation of servers $(r_1^*, \ldots, r_N^*) \in R$ by solving the optimization problem with $\sum_{i=1}^{N} g_i(r_i, u_i)$ substituted in Eq. (3). Assume for clarity of presentation that $f_i(\cdot)$ and $h(\cdot)$ are linear functions with slopes of unity. Then the router variables for each AE_i are obtained by minimizing the expected response time within AE_i, and the CM allocates servers among the set of AE_i in order to minimize the overall sum of the corresponding expected response times.

The optimization problem of each AM_i has been considered in [4] within the context of closed-form approximations based on heavy-traffic stochastic-process limits [3] to accurately model the per-server response time processes in each AE_i under general conditions in an on-

line fashion. We therefore exploit the results derived in [4] showing that the RHS of (2) for every AE_i is given by

$$\min_{x_i} \sum_{S_j \in r_i} \left[\frac{1}{\mu_{i,j}} + \frac{\lambda^2 \sigma_A^2 x_{i,j} + 1 - x_{i,j} + \mathcal{C}^2_{S_{i,j}}}{2(\mu_{i,j} - \lambda x_{i,j})} \right] x_{i,j},$$

s.t. $\sum_{S_j \in r_i} x_{i,j} = 1$, $x_{i,j} \geq 0$, $\lambda x_{i,j} < \mu_{i,j}$, where $\mu_{i,j}^{-1}$ and $\mathcal{C}^2_{S_{i,j}}$ are the mean and coefficient of variation of the service time process on server $S_j \in r_i$, λ^{-1} and σ_A^2 are the mean and variance of the overall interarrival time process for AE_i, and $x_{i,j}$ is the proportional weight for routing requests to server $S_j \in r_i$. The corresponding CM problem consists of solving for the set of servers $(r_1^*, \ldots, r_N^*) \in R$ in the following optimization problem:

$$\min_{r_i} \min_{x_i} \quad \sum_{i=1}^{N} \sum_{S_j \in r_i} \left[\frac{1}{\mu_{i,j}} + \frac{\lambda^2 \sigma_A^2 x_{i,j} + 1 - x_{i,j} + \mathcal{C}^2_{S_{i,j}}}{2(\mu_{i,j} - \lambda x_{i,j})} \right] x_{i,j}.$$

We have implemented the foregoing example and have conducted many numerical experiments. Using this framework, we find that even though the total amount of computation is larger for the decentralized approach, the optimization is distributed and the work performed by the CM is in general less than having the CM perform centralized global optimization.

6 Additional Fundamentals

The foregoing mathematical framework makes it possible for us to further explore several fundamental research issues concerning the decentralized approach to designing and implementing complex information systems with self-$*$ properties.

The control, decision making and optimization mechanisms cannot always be continuous (due to granularity) and may include some time-delay dependencies. One of the reasons is that any change requires time and resources. For example, switching a server from one application environment to another even if done in the same physical domain requires some clean-up and quarantine time (due to, e.g., privacy restrictions in SLAs). The time delays can also be caused by different time scales of the workloads as well as the operations of several applications within an environment. It is known that even very simple (e.g., linear) models which are only piecewise continuous or contain a feedback element may exhibit chaotic (in the sense of difficult to predict and qualitatively very sensitive to initial or control conditions) behavior. This chaotic behavior may appear in some regimes of parameters, however the sets of vulnerable parameters may also be very complex – excluding an envelope (e.g., closure or convex hull) of them might exclude an overly large portion of the parameter space.

In special cases chaos can be controllable, for example many stochastically stable systems exhibit individual chaotic trajectories, but with very well behaved distributions or moments. The transitions from a deterministic regime, where all trajectories are predictable at all times, to a stochastic regime, where most of the trajectories are predictable over long intervals of time, may go through all kinds of uncontrollable evolutions.

It is therefore essential for any given control system to determine the types of possible asymptotic behavior, the stability of such behavior under small perturbations of the system (a robustness question), and to conceive of mechanisms exposing the type of behavior the system is currently in.

Some of the research questions that we are currently pursuing include:

1. What is the stability of the algorithm when each AE_i operates at a different time scale and optimizes Eq. (2) at different speed and accuracy?

2. What is the potential for and impact of phase transitions and chaotic behavior in the decentralized system?

3. How does stochasticity and the time-varying nature of the external variables u_i affect the optimality and efficiency of the optimization algorithm?

4. How is the computational complexity and efficiency different for an algorithm computing h_c in Eq. (1) and for an algorithm computing h_d in Eq. (3)? Is the global minimum (versus a local minimum) harder to obtain in Eq. (3) than in Eq. (1) or vice versa?

References

[1] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[2] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust-Region Methods*. SIAM, 2000.

[3] D. Gamarnik, Y. Lu, and M. S. Squillante. Foundations of stochastic modeling and analysis for self-$*$ properties in autonomic computing systems. Technical report, IBM Research Division, March 2004.

[4] X. Guo, Y. Lu, and M. S. Squillante. Optimal stochastic routing in distributed parallel queues. Technical report, IBM Research Division, November 2003.

[5] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–52, 2003.

[6] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das. Utility functions in autonomic systems. Technical report, IBM Research Division, January 2004.