Self-Organizing Control in Planetary-Scale Computing[1]

Artur Andrzejak, Sven Graupner, Vadim Kotov, Holger Trinks
Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA
{*firstname_lastname*}@hp.com

Abstract

The explosion in globally connected devices, computers, and services ultimately evolves into very large-scale, inter-connected systems approaching a new era of *planetary-scale computing*. The backbone of planetary-scale computing is a global network of data centers. The paper provides an overview of research at HP Laboratories exploring new approaches under such a scenario based on HP's virtual data center platform enabling large-scale distributed virtual data center environments. Our approach is founded in a service-centric system view, and we explore agent technology for resource control, system organization, and balancing resource demand and supply.

1 Introduction

Planetary-scale computing becomes increasingly service-oriented: groups of interoperable applications provide functions or services on request to other groups of applications and to users in the end. *Services* encapsulate and isolate functionality from underlying application instances, which might be changed, replaced, or moved without affecting the use of a service. Service customers interact with services as black boxes rather than being tightly coupled with individual applications. As applications can be grouped, wrapped and encapsulated providing services, resources can be treated similarly providing *virtual resources*.

We refer to this paradigm as *service-centric* computing [1]. The provided degree of abstraction and separation is ultimately necessary in systems of planetary scale. A notion of *layers* allows partitioning the space where services and virtual resource of equivalent granularity can be correlated. Section 3 overviews these notions.

The background for our research is provided by new opportunities that HP's next generation *virtual data center platform* offers. Section 4 gives an insight into basic technical principles and explains the potential arising from the capability of total resource virtualization in order to create planetary-scale virtual data center environments where explicit support is provided for enabling applications to be grouped and encapsulated (wrapped) as services. As services are expected to be always on, they also need to supply sufficient quantity meeting demand and quality in an economic manner. Better than best-effort *demand and supply control* is inevitable for planetary-scale systems. We describe an *agent-based approach* in section 7.

2 General Approach

Traditional system management is facing limits when anticipating scale and reactiveness requirements due to:

- *too many* managed objects,
- of *large heterogeneity*, and
- of *low system level*.

Simple and reactive autonomous control is needed to meet the planetary-scale management challenge. The problem we are discussing in this paper is how to arrange a simple and efficient control for complex systems. By efficiency of the control we mean its ability to provide a desired balance in the usage of resources with simultaneous providing quality of services. The main task is to keep the system in balance, providing good resource utilization and ensuring the quality of services to the end-user. We are not addressing other aspects such as security.

The main technical hurdle is the large scale combined with time constraints. Our general approach is to attack this problem simultaneously from several directions:

- divide the overall control space into interacting *control layers* where services and virtual resources of equivalent granularity can be viewed appropriately,
- reduce the number of controlled objects by making them of *higher granularity*, uniform and simple,
- represent both controlled systems and control itself as *uniform, recursive* structures,
- use principles and mechanisms of partially distributed autonomous *self-control*, and
- exploit the *legacy* of existing monitoring and management systems such as HP OpenView.

[1] submission to IEEE International Symposium on Cluster Computing and the Grid (CCGrid), Berlin, May 21-24, 2002.

Stratification of control

Too many objects and aspects of large-scale systems are controlled today. By using a divide-and-conquer strategy, we stratify the control space into layers that address those tasks that are specific for different levels of abstraction. Currently, we consider two control layers: the lower-level *resource control* and higher-level *service control*.

The resource control manages physical resources, providing the service control with virtual resources. A physical resource is a device that has some absolute temporal or/and spatial computing parameters. A virtual resource is a set of parameters that are mapped onto physical resources of a device or a collection of devices.

Service control allocates the virtual resources among groups of applications. It monitors and controls the balance between the applications demands of virtual resources and the physical resources capacities. The focus of this paper is exclusively on service control.

Higher granularity

A natural way to reduce the number of controlled objects is to increase their granularity and simplify them by raising their level of abstraction. They should provide a clean separation between individual system components and their global role, visibility and behavior within the overall system organization. This separation, together with the higher-level abstractions, enhances controllability and leads to a system representation that is more tractable and more amenable to global analysis and control.

We propose to augment controlled systems with new types of objects of higher system granularity (without any modifications of underlying applications and computing resources). These objects serve as control points for groups of applications or groups of resources, which they represent in the control system. Their task is to report collective characteristics and to control collective behavior of the represented groups. Such control objects are introduced in Section 3.

Uniform recursive structure

The control system itself is potentially a complex system. It is a question how to build an efficient control infrastructure that is simple, flexible and yet easily adaptable to a large variety of types of systems. We use the same type of high-level representation for both controlled system and the control infrastructure in a hierarchical tree-like structure. It is also recursive in the sense that controlling objects are controlled themselves by other controllers.

Self-control

Self-control means autonomous self-analysis and self-governing of system parameters and states. Flexible mechanisms for coordination among entities in the control system (agents) are required that can implement centralized, partially distributed, and fully distributed models. Different mechanisms and algorithms should be used for different scale and reactivity requirements, but their number should be small and they will be automatically selected by the control infrastructure.

Legacy management

Legacy and customized monitoring and management systems should be used where it is appropriate to perform sensor and actuator tasks.

3 Service-Centric Organization

The service-centric view reduces the amount of information we want to monitor, the number of objects that are controlled, and complexity of these objects [2].

Our solution is to instrument the systems with control objects providing control points. Such control points are:

- *(core) services* that represent some set of applications implementing some tasks on request of other services (or users, in the end),
- *(virtual) servers* that represent virtual resources provided to services.

Services are embodiments of the demand and servers are embodiments of the capacity in a demand-capacity control model. Systems are viewed by the system control as a *service infrastructure* that consists of virtual servers hosting services. Aggregating services and servers forms hierarchical structures that support both abstraction and refinement needed for controlling large-scale systems.

Control points are described by *descriptor* XML documents that contain the externally visible information used for incorporating them into a control infrastructure.

The service infrastructure [2] includes:

- *deployment mechanisms* of service entities,
- *XML-based messaging* among server and service control objects,

[2] The service-centric view of systems was introduced initially as an abstraction that makes it possible to design, model, and analyze large-scale communicating systems [2]. We found that the same concepts are useful in the control context what can be referred to more widely as *service-centric system organization.*

- *repositories*, accumulating templates of servers and services for subsequent deployment,
- *directories*, containing actual information about service entities and their deployment.

To deploy conveniently aggregated entities, a recursive schema of "self-deployment" has been proposed, in which deployed aggregated entities deploy their children [3].

4 Virtual Data Center

Large customers, such as big corporations or e-service providers, profit from or are required spanning of their data and applications over several data centers, potentially owned by different organizations, combining the resources of these centers. Smaller customers tend to use only part of the resources and services of a data center to which they outsourced their IT infrastructure. Combination of these options may occur in the general case. Customers form their private virtual data centers that provide for them resource consolidation and location independence. The virtual centers also extend the customer's capability to optimize the center for their business needs and for particular workloads hiding the actual platform architecture and providing a much simpler and convenient virtual architecture called a *Virtual Data Center (VDC)* [4] transparently bridging traditional geographic and organizational data center boundaries.

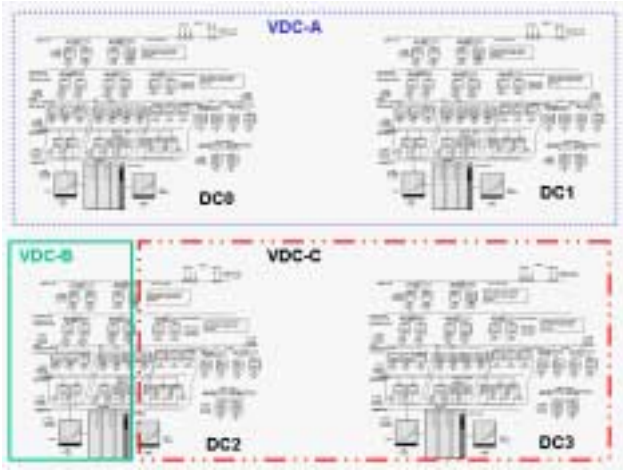

Figure 1: Virtual Data Centers (VDC)

This approach assumes some support by a platform where unmodified application systems can run and yet be controlled in the described ways. HP's virtual data center platform [5] provides total resource virtualization and expands connection fabrics from LAN to the WAN scope. Total resource virtualization is achieved by programming the virtually wiring of physical resources (machines and storage).

Customer applications, programs and data, management systems, even operating systems, are installed, managed and maintained within a customer's virtual environment rather than on physical machines. The control system organizes the mapping of customer's virtualized resources onto physical resources. Since current applications, management systems and operating systems cannot be changed, resource virtualization has to be transparent, sitting underneath the abstractions operating systems expect as the lowest layer in system stacks similarly to virtual machines, but here in a data center context. Three types of resources are virtualized:

- processing; machines with CPU's and memory,
- storage with images containing file systems with operating systems, application software and data, configurations, etc., and
- a programmable, switched fabric between processing and storage units based on programmable Fiber-Channel (FC) and/or Ultra-Fast SCSI links and switched G-Ethernet among the processing machines and for the connection to the outside.

All connector fabrics, switched Ethernet and Fiber Channel, are programmable by the VDC control system. This programmability provides resource virtualization similarly as programmability of page tables provides memory virtualization in operating systems. Processing elements may be dynamically mapped into a customer or service domain with their customer-defined identity (IP address). Fiber Channel connectivity allows the programmability of which portions of storage appear as disk images on FC or SCSI interfaces in processing elements containing the operating system being booted and file systems being mounted later. It is assumed that customer state is entirely maintained in the storage system. Processing and storage elements are regular machines and systems currently available from HP or other vendors. The control system performing the mapping of virtual to physical resources and maintaining customer states in form of portions of disk images is the key element for operating such an environment.

5 Controlling the Demand and Supply Balance

In order to keep demand and supplied capacity in balance, the control system needs to optimize the placement of the services on servers, monitor the balance, and then eventually re-deploy them if the balance is lost.

Arising questions are: What metrics can be used to describe the resource demand and supplied capacity? Is it

possible to use one "generic" metric? How complex should it be?

We use simple parameter sets to express processing, storage and communication resources in a format normalized to a chosen base unit specific to a given environment, see [2] for more detail.

Given a VDC environment, decision-making refers to keeping resource demands and supply capacities in balance as a permanent process and as part of VDC's control system. We assume that demand will drive this control. Demand may occur sporadically, unpredictably or may follow known patterns. Demand may strongly fluctuate or may change rather steadily. Four basic controls are applicable:

1. adding capacity transparently to given resources; an example might be adding disk space to a file system,
2. re-directing demand to locations where resources are available; load balancing is an example here,
3. finding a better arrangement or placement of demands on supplied capacities reducing the overall demand; an example is bringing data closer to locations where it is accessed in order to keep traffic local and by thus reducing or avoiding traffic in the overall system.
4. rejecting demands above certain thresholds known as admission control.

A combination of all these approaches is typically chosen. Decisions need to be made in the control system how to react on changing demand conditions. Demand-driven resource control means permanently monitoring the system and comparing observed with expected behavior in order to detect misbalances between current resource capacities and current resource demands. Decisions may then be further evaluated and refined and finally be implemented in the system, manually or partially automated. [2] shows a prior model-based approach we investigated in the System Factory framework.

6 Agent-Based Control Infrastructure

The proposed control infrastructure is itself organized as a service-centric system built of special monitoring and control *agent-services*. They may be aggregated and communicate with each other as any other services. This supports recursive control, as services may control other services and are being controlled by other services.

To manage automatically a large number of services and servers in a changing distributed environment, the control system should be distributed as well. The main control functions, such as monitoring, decision-making, and control execution, are delegated to *agent-services* of the appropriate functionality and multiplicity. They may form hierarchical aggregations and may, in their turn, be controlled by higher-ordered control entities. This results in a *distributed recursive schema of control*. Monitoring is also distributed among various levels of granularity, and control decisions are also decentralized.

The agent- and service-based control

- collects, monitors the information needed for control,correlates services' demands with current servers' capacities at appropriate levels of granularity, and
- uses smart algorithms to *predict*, *optimize* and *decide* about arrangements keeping systems in balance.

7 Decision-Making

We now focus on decision-making for managing the resource demand and supplied capacity. Optimal placement of services on servers is the primary factor for:

- economic utilization of the underlying resources,
- preventing overloading servers or the communication infrastructure,
- keeping resource utilization and response time in balance, and
- high-availability and fault-tolerance.

The biggest challenge is to find such algorithms that are both reactive and deliver high-quality solutions for the control scale we are dealing with. In practice, the responsiveness of an algorithm must be traded against the quality of a solution. Thus, responsiveness constitutes one parameter of the design space. Another parameter is the type of the control system, ranging from centralized to completely distributed. Since it is not realistic to find one algorithm, which can be parameterized in both dimensions, we look at several approaches covering most of the design space.

One approach we pursue is a centralized heuristic algorithm based on integer programming. This algorithm yields high-quality solutions but at a cost of longer running time. For improved responsiveness, we explore agent-based and distributed algorithms described below. Such algorithms are composed of several simple parts, each potentially residing in a separate control service. They communicate with each other directly or indirectly in order to obtain an approximate solution. Each part has, in general, only partial knowledge of the whole system. This facilitates scalability of such approaches. Furthermore, failure of any of the parts does not make the overall algorithm fail. The work includes the design of a

hierarchical overlay control structure, which determines how the parts of the distributed algorithms are placed and how they communicate with one another.

One agent-based approach is based on the "ant-based" control [6], [7]. This fully distributed algorithm has medium responsiveness and can be used for periodical reassignments of services onto servers.

As an alternative approach, we evaluate an agent system based on a paradigm known as Broadcast of Local Eligibility (BLE), used for coordination of robot teams [8]. This partially distributed algorithm allows faster rebalancing of the managed services for the price of a possibly lower-quality assignment.

Optimization Objectives

As discussed in the beginning of this section, the goals for optimal placement might vary in general. Therefore, the following algorithms are designed to be generic enough to support new objectives without fundamental changes. However, we focus on only few aspects to be achieved by the optimization. These are:

1. Balancing the server load such that the utilization of each server is in a desired range.
2. Placing services in such a way that communication demand among them does not exceed the capacity of the links between the hosting servers.
3. Minimizing overall network traffic aiming to place services with high traffic among each other on nearby servers (nearby in the sense of low number of hops).

Ant-Based Control Algorithm

In the classical Ant Colony Optimization [6] the path taken by an ant on its way between objects (e.g. cities in the Traveling Salesman Problem) represents a possible solution to the optimization problem. In our case, the objects would be both servers and services, and the alternating path would represent an assignment of services to servers. However, this approach is centralized and not really scalable for the following reasons:

1. The ant must "remember" the whole path it has taken; this information might become very large in the end.
2. The ant must visit all objects on its tour. In a changing system, this could pose a serious drawback.
3. Finally, each solution (path) must be evaluated against others. This requires central knowledge.

Due to these reasons, we evaluate another approach resembling the ant-based control for network management described in [7].

In our system, for each service *s* to be placed on a new server due to an overload condition or a new service has arrived, an ant ("agent") is created. This ant knows the requirement attributes of the service it represents, such as processing and storage requirements. It also carries a list of services cooperating with *s*, together with their communication requirement attributes.

The ant then travels from one server to another choosing the servers along the path based on a probability computed as described below. The ant then finally makes a decision based on the knowledge it has accumulated on its travel on which server the service will be placed.

On each server, the ant evaluates the score of the server in respect to *s*. This score expresses how well the currently visited server is suitable for the placement of *s*. Once evaluated, a data structure of the current server called *multimark* is updated with the score of *s* and with flags indicating services cooperating with *s*. The multimarks of the nearby servers are updated as well, both with the score for the placement of a cooperating service and with related flags for *s*.

The score for a server in respect to *s* is computed by the following criteria:

- How well the server meets the requirement attributes of *s* depending on its recent utilization history?
- Can cooperating services be placed on nearby servers?
- What is the amount of the weighted traffic between the current server and the servers potentially hosting the cooperating services? The weight factors here are the distances between all server pairs.

Thus, the computed score represents fairly well the optimization objectives stated above.

The choice of an ant which server to visit next is based on the current utilization of the considered server (the probability decreases with higher utilization) as well as on the value of the multimark for *s* (or for one of the cooperating services, if the multimark for the targeted server has no entry for *s*). Here, we use the fact that multimark records contain both the score of the ants, which visited this server as well as information of the ants corresponding to its cooperating services.

The termination of the walk of an ant is determined by a parameter set upon its creation – the maximum number of servers to be visited. It gives us a partial control of the trade-off between responsiveness and the solution quality. Upon termination, the ant determines the server with the highest score from an internal (limited length) priority list. It then sends a message to the managing agent of this server with the suggestion to install *s* on it.

BLE-Based Control Algorithm

We adapt the concept of the Broadcast of Local Eligibility used for coordination of robots [8] for the placement of services. This concept can be used to create highly fault-tolerant and flexible frameworks for coordination of systems of agents. However, the originally proposed framework has a drawback of limited scalability. To overcome this problem, we use a hierarchical control structure discussed below.

We consider a cluster of servers with a distinguished server called *cluster head*. Each member of the cluster has the ability to broadcast a message to all other members of the cluster. (This can be done either directly or via the cluster head). The placement of services in this cluster is periodically re-evaluated by arbitration between peer servers in so-called decision cycles. The time between two cycles is determined by the required responsiveness to fluctuations in server utilization and by the induced communication between cluster members.

In each decision cycle, the following actions take place:

1. Each server broadcasts the list of services it hosts with all new emerged services, and simultaneously collects a list of all services in the cluster.
2. Each server evaluates its own suitability to host each service and sorts the list according to the computed score. The criteria are similar to those for the ant-based control system. In addition, a service already running on a server highly increases the score.
3. Each server broadcasts a list ordered by scores of those services the server can host simultaneously without exceeding its capacity.
4. When a server receives a score list from a peer, it compares the score with its own score for a service. Each server now knows whether it is the most eligible one for hosting a particular service.
5. The changes in the service placement are executed. Notice that each server knows already whether it has to install new or remove current services. In addition, the cluster head compares the initial list of the services with those, which will be hosted at the end of this decision cycle. The remaining services are passed on to the next hierarchy level as explained below.

Obviously, the scalability of this approach is limited by the size of the cluster, the communication capacity in the cluster and the processing capacity of the cluster head.

We propose a following hierarchical approach to extend the scalability. Basically, the cluster heads of the clusters at level *k* are treated as "normal" members of a cluster of level *k+1*. However, they compete only for services, which could not be installed in their own cluster (see 5. above). After a decision round in the cluster of level *k+1*, these pending services are possibly moved to another peer, which is a cluster head for a cluster of level *k*. (The cluster head evaluates the eligibility of the servers in its own cluster, not its own eligibility). In the cluster of level *k*, these services become part of the list of services to be installed and participate in the normal decision cycle.

The cluster size is essential for the balance between the responsiveness of the system and flexibility. Identifying a correct hierarchical structure can be done similarly to clustering algorithms used in sensor networks [9].

Decentralized control algorithms appear to be promising for decision-making in large-scale virtual data centers as part of their control systems.

References

[1] Kotov, V.: *Towards Service-Centric System Organization*, HP Labs Technical Report, HPL-2001-54, March 2001.

[2] Graupner, S., Kotov, V., Trinks, H.: *A Framework for Analyzing and Organizing Complex Systems*, Proceedings of the 7th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2001), pp. 155-165, Skövde, Sweden, June 11-13, 2001.

[3] Graupner, S., Kotov, V., Trinks., H.: *Recursive Deployment of Management Agents in Planetary-scale Control Systems*, HP Labs Technical Report, to be published in December 2001.

[4] Kotov, V.: *On Virtual Data Centers and Their Operating Environments*, HP Labs Technical Report[3], HPL-2001-44, March 2001.

[5] Rolia, J., Singhal, S., Friedrich, R.: *Adaptive Data Centers*, Proceedings of SSGRR 2000 Computer and eBusiness Conference, L'Aquila, Italy, August 2000.

[6] Dorigo, M., Maniezzo, V., Colorni, A.: *The Ant System: Optimization by a Colony of Cooperating Agents.* IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41, 1996.

[7] Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: *Ants for Load Balancing in Telecommunications Networks*, Adaptive Behavior 2:169-207, 1996.

[8] Werger, B. B., Matarić, M.: *From Insect to Internet: Situated Control for Networked Robot Teams*, to appear in Annals of Mathematics and Artificial Intelligence, 2000.

[9] Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: *Next century challenges: Scalable coordination in sensor networks*, Proceedings of MOBICOM, pp. 263-270, Seattle, USA, August 1999.

[10] *HP Utility Data Center*, http://www.hp.com/go/hpudc, http://www.hp.com/go/always-on, November 2001.

[3] HPL-TR are available: http://lib.hpl.hp.com/techpubs.