Autonomic Management of Distributed Systems using Online Clustering

Andres Quiroz, Manish Parashar, Ivan Rodero
NSF Center for Autonomic Computing
Department of Electrical & Computer Engineering
Rutgers, The State University of New Jersey
Email: {aquirozh,parashar,irodero}@cac.rutgers.edu

I. Introduction

Distributed computational infrastructures, as well as the applications and services that they support, are increasingly becoming an integral part of society and affecting every aspect of life. As a result, ensuring their efficient and robust operation is critical. However, the scale and overall complexity of these systems is growing at an alarming rate (current data centers contain tens to hundreds of thousands of computing and storage devices running complex applications), making the management of these systems extremely challenging and rapidly exceeding human capability. Furthermore, these systems require simultaneous management along multiple dimensions, including performance, quality of service, power, and reliability.

The large quantities of distributed system data, in the form of user and component interaction and status events, contain meaningful information that can be used to infer the states of different components or of the system as a whole. Accurate and timely knowledge of these states is essential for verifying the correctness and efficiency of the operation of the system, as well as for discovering specific situations of interest, such as anomalies or faults, that require the application of appropriate management actions.

Autonomic systems/applications must therefore be able to effectively process the large amounts of distributed data and to characterize operational states in a robust, accurate and timely manner. Although highly accurate, centralized approaches for distributed system management are infeasible in general because of the costs of centralization in terms of infrastructure, fault tolerance, and responsiveness. Since data is naturally distributed and the collective computing power of networked components (ranging from sensor and device networks to supercomputer clusters and multi-organization grids) is enough to be harnessed for value added, system-level services, online and decentralized approaches for monitoring, data analysis, and self-management are not only feasible, but also quite attractive.

This work is based on the premise of realizing online data analysis, exploiting the collective computing resources of distributed systems for supporting autonomic management capabilities. Specifically, we explore online clustering as a data analysis mechanism and infrastructure and apply it to the following autonomic management problems: 1) System profiling and outlier detection from distributed data, and 2) definition, autonomic adaptation, and application of management policies.

In the following sections, we will discuss our approach for clustering, as well as specific challenges of the aforementioned problems and the way that online clustering is used to address them. We will also describe specific scenarios to give context to the different problems and to demonstrate the usefulness of the proposed approach. First, however, we will briefly describe our online clustering approach and infrastructure.

II. Online Clustering

Consider events in a distributed system to be represented as *points* in one or more multidimensional spaces. Each dimension in a particular space, referred to as an *information space*, corresponds to an attribute that describes part of the behavior or state of a user or other system component. The location of a point within the space is determined by the values for each of the attributes of which the space is composed.

Our approach for cluster detection is based on evaluating the relative density of points within the information space. In order to evaluate point density, the information space is divided into regions that are analyzed separately. If the total number of points in the information space is known, then a baseline density of a uniform distribution of points can be calculated and used to estimate an expected number of points per region. Clusters are recognized within a region if the region has a relatively larger point count than this expected value. Conversely, if the point count is smaller than expected, then these points may potentially be outliers or isolated points. These concepts are illustrated in Figure 1.

The approach described above lends itself to a decentralized implementation because each region is assigned to a particular *processing node*. Also, it can be done online because points are routed to processing nodes as they are produced, and processing nodes analyze the points within their region independently, performing a very lightweight computation (basically comparing point counts), and only communicating with nodes responsible for adjoining regions to deal with boundary conditions and for aggregating cluster data. We assume that the information space (or spaces) is (are) predefined and globally known; however, the assignment of regions to nodes need not

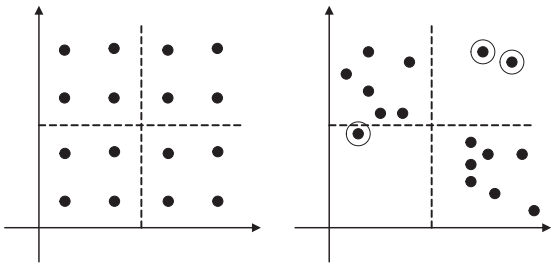

Fig. 1. (a) Uniform distribution of data points in the information space. (b) Point clustering and anomalies among regions; the circled points are potential anomalies.

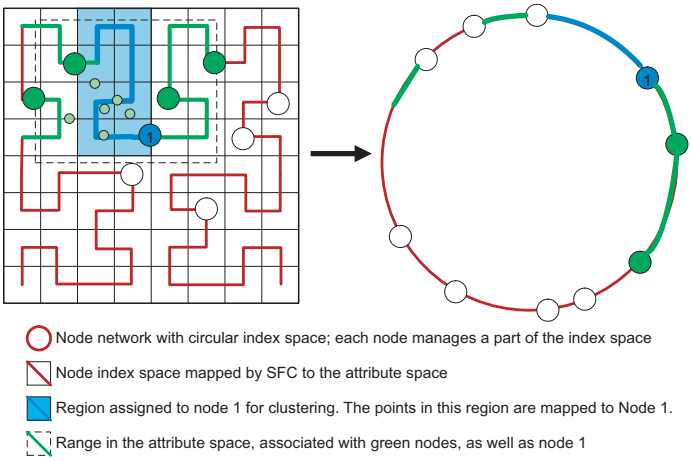

Fig. 2. Mapping from attribute space to node index space. Matching of cluster points and ranges can be done because intersecting regions are guaranteed to map to a non-empty set of common nodes, such as node 1 in the figure.

be static, and can thus be adapted to the arrival and departure of processing nodes in the network, as long as a mechanism exists to map data points to the node responsible for the region in which the point is located.

The exchange of job requests is supported by a robust content-based messaging substrate [1], [2], which also enables the partitioning of the information space into regions by implementing a dynamic mapping of points to processing nodes. The messaging substrate is responsible for getting the information used by the clustering analysis to the distributed processing nodes in a scalable fashion. The substrate essentially consists of a content-based Distributed Hashtable (DHT) that uses a Peano-Hilbert space-filling curve [3] as a locality preserving hash function. Content-based DHTs provide an interface that allows network nodes to be addressed by attribute-value pairs, so that, given a point with a value for each dimension (attribute), the DHT can find a route to the node responsible for the region containing that point. Additionally, our messaging substrate is able to resolve multidimensional range queries (in which a range of values is given for each dimension), guaranteeing the discovery of every node whose region intersects with the range of the query. Figure 3 shows how points, clusters, and ranges are mapped by the DHT to processing nodes connected by a ring overlay. A detailed analysis of the performance of content-based routing achieved by this substrate can be found in [1].

III. Profiling and Outlier Detection

A profile characterizes the behavior and/or interactions of a managed entity or collection of entities within the system within some period of time. It is based on the identification of patterns in this behavior, which are manifested through clusters of events corresponding to these entities. In other words, if the behavior of some entity or collection of entities exhibits some similarity or consistent property over time, then the events corresponding to that behavior will tend to cluster in the information space in some way.

For each managed entity in the system, it is therefore possible to maintain a profile composed of profile elements. These correspond to either cluster descriptors or to a set of individual events that represent spurious or abnormal behavior. It is important to maintain outliers within a profile because they indicate noteworthy behavior that can be used to raise alerts and can either be reinforced over time or eliminated if the behavior is not repeated.

Behavior evolves over time and, consequently, profiles must be updated accordingly. The effectiveness of profiling within a distributed system not only depends on having a (clustering) mechanism that is able to produce and update clusters at runtime fast enough to adapt to the dynamism of behavior within the system, but also on employing further pattern recognition and learning techniques that can be used to interpret profile data. For example, cluster centroids can be clustered over time so that similar centroids, corresponding to a continuing trend of the entity's behavior, can be merged into a reinforced profile of the entity. Similarly, any outliers found in the data can be clustered over time to recognize the frequencies of certain abnormal events or possibly patterns that occur in a different time frame than that of the regular clustering window.

IV. Dynamic Policy Generation

Policies are a powerful means of expressing high-level, goal oriented parameters that manage the behavior of systems and users, and are thus valuable tools for autonomic management. However, the autonomic management of policies is in itself a challenging problem. Polices are typically defined with static constraint thresholds and are either associated with specific states of the managed entities or applied reactively in response to feedback from events or actions. This limits their applicability to situations where the appropriate management actions depend on dynamic system properties, which require adapting policy application thresholds and parameters without modifying absolute policy definition constraints.

The analysis and characterization of interaction and feedback events during system operation using online clustering can support autonomic policy adaptation and proactive policy application by establishing a mapping between events (clusters) in a space of attributes used for policy definition (policy definition space) and events in a space of attributes subject

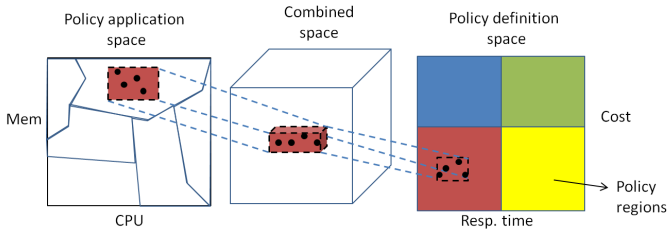

Fig. 3. Clustering as a tool to discover parts of the mapping between policy spaces

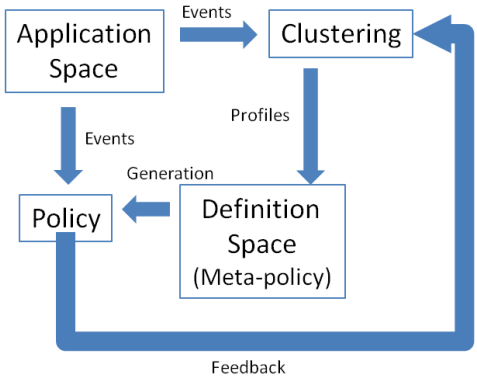

Fig. 4. Overview of our approach. Note that policies are derived from meta-policies using clustering results and are applied to events in the application space before feedback is available.

to management actions (policy application space). Figure 3 illustrates this concept.

The value of this approach lies in the difficulty inherent in distributed system of defining an application region (constraints or thresholds) of a policy, because the relation between the event attributes that are being managed and the system states to which management actions are attached is not known and/or is dynamic. In our approach, we refer to policies in the definition space as meta-policies, because a meta-policy is actually a specification of when and how a dynamically generated policy should be applied. In order to derive policies from meta-policies one must approximate the mapping of attribute values between the two policy spaces. Our approach, illustrated in Figure 4, is to correlate both spaces by observing the feedback attributes for a sequence of monitored events and finding temporary mappings by finding clusters in a combined space of all attributes. These clusters determine the dynamic boundaries of policies in the application space.

V. Application Scenarios

A. Usage Control in a Device Network

We focus on a scenario from an enterprise domain, where multiple users interact with numerous document services deployed on multi-function devices spread across the organization. In this type of device network, it is hard to come up with consistent resource quotas and access privileges a priori because of the following: a) Most users have their own unique time-of-day dependent usage profile which deviates

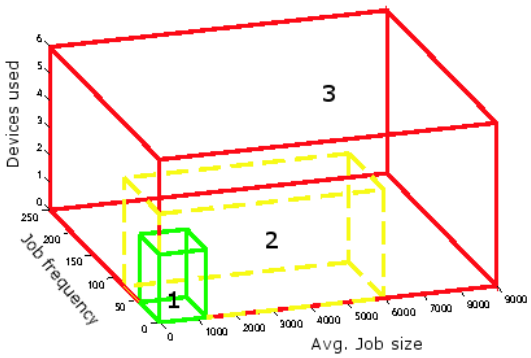

Fig. 5. Representations of the meta-policy regions as a priority mapping within the attribute space

significantly from other users; and b) Since these devices are internal to the organization, most users of services have more than an adequate (in some other cases a very restricted) set of privileges assigned to them, causing cost inefficiencies and security threats. As a result, administering polices and accounting for changing user behavior is difficult in this domain.

We can define a meta-policy for this scenario by identifying definition space attributes according to the management goals. In this case, the system goal is having a fair and balanced use of resources in the enterprise. The attributes that describe the usage trends of the system over time (obtainable from job requests) are the average job size of a user within a given time period, the number of distinct devices printed to by the user, and the job submission frequency of the user in the period of time. A system administrator, based on experience, can set regions of interest in the attribute space that indicate the perceived fairness of situations in which clusters of particular characteristics fall within these regions. Possible regions for this scenario are shown in Figure 5.

A simple meta-policy that can be defined based on these attributes is one that assigns priorities to jobs according to usage profiles and device utilization. For example, jobs from low impact users (infrequent small jobs) to underloaded devices can be given a greater priority than jobs from higher impact users and/or to more highly loaded devices. This meta-policy can be realized by associating increasing priorities from the origin to the regions of Figure 5.

The application space in this scenario is composed of request attributes of individual job requests. Namely, the attributes are job size and the device to which the request is sent. Note that these two attributes by themselves are not enough to define a meaningful policy for priority based on usage, but are the only ones available when the job is submitted. By applying the clustering-based dynamic policy approach, it is possible, for example, to obtain different usage profiles online for different times of day, and to apply these profiles for policy generation. Once policies are generated for particular users, the system will be able to proactively assign

priorities to jobs directly from the application space attributes.

We have evaluated this scenario using data obtained from the monitoring of a system of distributed multi-function document processing devices that serve about 200 users in an office environment. Our results demonstrate the dynamism of policy generation using our approach by producing significantly different priority distributions for incoming jobs when policies were generated for morning and evening user profiles.

B. Autonomic Workload and VM Provisioning

In a cloud environment, executing application requests on underlying Grid resources consists of two key steps. The first, which we call VM Provisioning, consists of creating VM instances to host each application request, matching the specific characteristics and requirements of the request. The second step is mapping and scheduling these requests onto distributed physical resources (Resource Provisioning). While much work to date has been dedicated to the resource provisioning problem, under-utilization of resources resulting from inappropriate VM provisioning will still occur.

In order to improve resource utilization, it is necessary to reduce overprovisioning at two levels. To reduce overprovisioning caused by the difference between the virtual resources allocated to VM instances and those requested by individual jobs, online clustering can be used to efficiently characterize dynamic (rather than generic) classes of resource requirements that can be used for proactive VM provisioning. To address the inaccuracies in client resource requests that lead to overprovisioning, dynamic policies can be used to transform these requests to better conform to the usage trends observed in client and application profiles.

Profiling of VM classes using clustering not only leads to decreases in overprovisioning, but also to the possibility of using existing classes to proactively provision new VM instances for incoming jobs. This is because the job profiles found through clustering characterize the expected resource configurations required by incoming jobs. As an example, Figure 6 shows a two-dimensional information space with three initial clusters (shown as squares). The top right-hand corner of these squares represents the resource values that characterize the class of jobs belonging to that cluster. Any job that falls within the regions represented by the dotted lines (like the star in the figure) can be automatically assigned to the closest pre-provisioned VM type. The circle is a cluster that is discovered in a subsequent time window. This cluster can be used to create and provision a new VM type. The **x** is a request that falls outside existing regions (outlier) and represents a job whose requirements exceed the currently provisioned resources. In this case, the cost of reactively creating a new VM type is also incurred. In [4], we showed how VM provisioning with this approach can achieve an over 90% reduction in overprovisioning for CPU and memory in requests from a job trace from the Los Alamos National Labs (LANL) Connection Machine [5].

As mentioned earlier, meta-policies can be defined with respect to the slack in the requested vs consumed resources

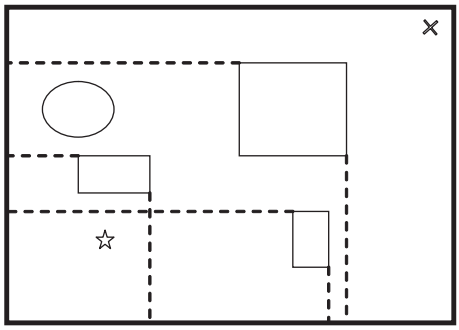

Fig. 6. Example information space and clustering results for dynamic workload provisioning.

of the jobs submitted to the data center in order to prevent over or underestimation of required resources and make sure that the incoming job requests are as accurate as possible. The policies that will be generated can actually modify the requested values contained in incoming jobs, before those jobs are serviced. This was applied to a job trace from the EGEE Grid Obervatory [6] with requests for CPU time and memory resources, and the transformations to the requests reduced the slack 25% for memory and up to 90% for CPU time. The difference in the reduction for the two resources was mostly due to the small amount of slack originally found in memory requests.

Clustering can be applied to the problem of resource provisioning as well, and we have begun exploring this from an energy perspective. The VM resource classes obtained from VM provisioning can also be used to classify jobs according to the level of resource consumption for different resource modules. For example, jobs can be classified as CPU-intensive, IO-intensive, network-intensive, etc., and this classification can be used to configure and possibly power down resource modules for different resource classes. We evaluated this approach based on simulations, using the workloads from the Grid Observatory and LANL. Compared to typical reactive or predefined provisioning, our approach achieves significant improvements in energy efficiency (around 15% on average) with an acceptable penalty in QoS (less than 5% in workload execution time).

REFERENCES

[1] C. Schmidt and M. Parashar, "Flexible information discovery in decentralized distributed systems," in *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12'03)*, 2003.

[2] A. Quiroz, N. Gnanasambandam, M. Parashar, and N. Sharma, "Robust clustering analysis for the management of self-monitoring distributed systems," *Journal of Cluster Computing*, vol. Online, pp. –, 2008.

[3] H. Sagan, *Space-Filling Curves*. Springer-Verlag, 1994.

[4] A. Quiroz, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds," in *Proceedings of the GRID Computing Conference (GRID 2009)*, Banff, Canada, October 2009.

[5] D. Feitelson, "The parallel workloads archive," http://www.cs.huji.ac.il/labs/parallel/workload/.

[6] "Grid observatory website," http://www.grid-observatory.org/.