CONDESA: A Framework for Controlling Data Distribution on Elastic Server Architectures

Juan M. Tirado, Daniel Higuero, Javier Garcia Blas, Florin Isaila, and Jesus Carretero

Abstract—Applications running in today's data centers show high workload variability. While seasonal patterns, trends and expected events may help building proactive resource allocation policies, this approach has to be complemented with adaptive strategies which should address unexpected events such as flash crowds and volume spikes. Additionally, the limitations of current I/O infrastructures in the face of dramatic increase of data generation require, the ability to build novel abstractions and models for robust decision making regarding data layout and data locality. In this work, we present CONDESA (CONtrolling Data distribution on Elastic Server Architectures), a framework for exploring adaptive data distribution strategies for elastic server architectures. To the best of our knowledge CONDESA is the first platform that permits to systematically study the interplay between five data related strategies: workload prediction, adaptive control of data distribution and server provisioning, adaptive data grouping, adaptive data placement, and adaptive system sizing. We demonstrate how CONDESA can be used for browsing the design space of adaptive data distribution policies. We show how prediction models can be compared in terms of overhead and accuracy. We evaluate the impact of change detection on prediction accuracy and how CONDESA can be used for choosing an adequate prediction horizon. We demonstrate how adaptive prediction can be used for sizing a server system. Finally, we show how prediction models, change detection strategies, and data placement policies can be combined and compared based on server utilization, load balance, data locality, over- and underprovisioning.

Index Terms—Prediction, adaptive control, data distribution, cloud

1 Introduction

The growing amount of information to be stored and processed in a data center and the high workload variability shown by Internet applications have increased the importance of efficient data management. Data placement has become one of the most important topics in data management due to its relevance for the desired data center operation and its direct impact on performance, elastic scalability, resource sharing, availability, and energy-efficiency. Controlling data layout is critical for feasibility of processing large data amounts, as the reorganization of large data sets is severely limited by the I/O infrastructure capabilities and must not negatively impact quality of service requirements [1]. However, controlling data layout has become an increasingly difficult task as the workloads of Internet applications show high variability due to factors such as periodic variations (seasonality), trends, expected and unexpected events. Traditionally, these variations have been addressed by overprovisioning the infrastructure, but this approach has been demonstrated to be costly and economically risky, as the peek volume is short-lived and the server utilization in normal traffic periods is between 10 percent and 50 percent [2].

The advent of cloud computing was a suitable match for the variable demand of Internet applications. Cloud computing enables elastic horizontal scalability of server infrastructures, which allows to dynamically allocate resources depending on demand and to pay only for the used resources. However, efficiently exploiting the dynamic resource allocation mechanisms offered by clouds strongly depends on understanding and controlling the dynamics of workloads and on reducing the data traffic inside the data center. Increasing scale and demand variations pose huge challenges on developing, deploying, and evaluating control mechanisms and policies for efficient resource allocation.

The main goal of our work is to simplify the process of designing and evaluating dynamic data distribution strategies for efficiently mapping the variable demand of applications on elastic infrastructures such as clouds. For achieving this goal, this paper proposes CONDESA (CONtrolling Data distribution on Elastic Server Architectures), a framework that leverages adaptive prediction and control techniques for facilitating the development of elastic data distribution policies for scalable applications with variable data workload patterns. Building elastic data distribution policies for highly variable demand is challenging, as changing workloads require data to be dynamically regrouped and redistributed in order to make an efficient use of resources and achieve economy of scale. When addressing this problem a designer works in a complex setup requiring to select and combine a series of complex techniques including:

1. building prediction models of workload variations at various granularities;

- *J.M. Tirado is with INRIA Rennes-Bretagne Atlantique, Campus Universitaire de Beualieu 35042 Rennes Cedex, France. E-mail: juan-manuel.tirado@inria.fr.*
- *D. Higuero, J. Garcia Blas, F. Isaila, and J. Carretero are with the Computer Architecture Group, Universidad Carlos III de Madrid, C.P. 28911, Leganes, Madrid, Spain. E-mail: {dhiguero, fjblas, florin, jcarrete}@arcos.inf.uc3m.es.*

Manuscript received 9 Oct. 2012; revised 10 July 2013; accepted 22 July 2013. Date of publication 20 Aug. 2013; date of current version 16 July 2014. Recommended for acceptance by J. Cao.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.197

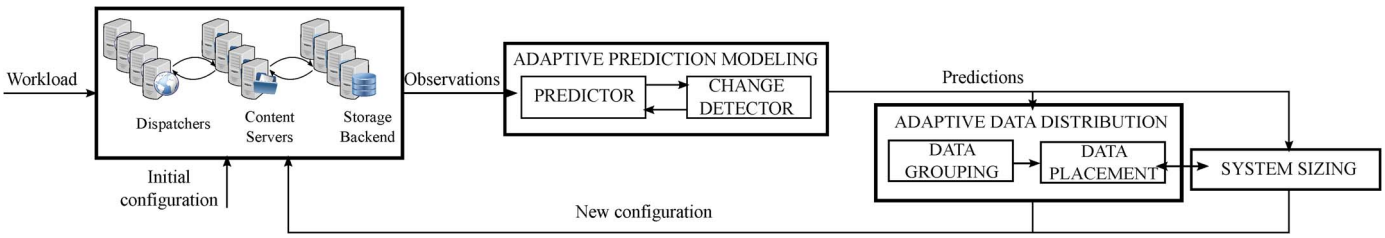

Fig. 1. CONDESA architecture.

2. adaptation to both permanent changes in workload patterns and unexpected workload spikes;
3. techniques for elastically scaling up and down the infrastructure depending on demand variation;
4. techniques for dynamically grouping data into allocation units called placement groups;
5. techniques for dynamically placing groups over servers or disks.

For evaluating the dynamics of an elastic infrastructure, CONDESA provides several metrics for estimating resource utilization, load balance, data locality, and modeling accuracy.

CONDESA framework emerges as an overarching approach of our previous work on dynamically provisioning elastic server infrastructures [3], [4], [5]. Beyond those publications this paper presents the following new contributions. First, CONDESA abstracts away the workload modeling, data grouping, and data placement strategies. Any time series prediction model, data grouping, and data placement strategies can be defined, integrated, and evaluated together with the other framework features. Second, CONDESA unifies local and global workload models into a hierarchy, allowing to take advantage of the tradeoff between global and local predictions. Third, CONDESA provides adaptability by automatic detection of inaccurate models and reactive model redefinition. Fourth, we introduce novel metrics for evaluating efficiency of dynamic data distribution and server allocation policies.

The remainder of this paper is structured as follows. Section 2 presents CONDESA architecture. Then, we discuss the main CONDESA components. Section 3 presents the adaptive prediction component. Section 4 discusses the system sizing module. Section 5 describes the adaptive data distribution module. Section 6 presents the data set used in the evaluation. Section 7 describes CONDESA metrics. Section 8 demonstrates a subset of analyses that can be performed with the CONDESA framework. We overview related work in Section 9, and finally, Section 10 concludes. Additional material is provided in a supplementary file consisting of six appendices which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.197.

2 CONDESA Architecture

CONDESA is a framework for evaluating adaptive and predictive data distribution techniques for elastic server infrastructures. CONDESA addresses three-tiers architectures [6], as they are the most common solutions for data-intensive Web-based applications. The three tiers of the model architecture are: dispatchers, content servers, and storage backend, as shown in the left-hand side of Fig. 1. The *dispatcher* tier receives application requests for storage objects (o_i) and redirects them to content servers based on a request distribution policy. The *content servers* return the demanded storage object to the application either from a local cache or from the *storage backend*.

The notations used in this paper are summarized in three tables shown in Appendix A available online: the input parameters in Table 1, the observed and predicted variables in Table 2, and the model evaluation criteria in Table 3.

CONDESA consists of three main components connected in a feedback loop with the controlled system as shown in Fig. 1:

- The *adaptive prediction modeling* component has two main tasks: 1) to define, monitor, modify, and adapt time series workload models; 2) to inform the adaptive data distribution component about model failures that could cause reactive resource provisioning actions. This component has as input the current workload at server level $(l_s(t))$, group level $(l_{g_k}(t))$, and global level $(l(t))$. The adaptive prediction modeling component consists of two interacting modules: a predictor module and a change detector module. The *predictor module* manages a extensible library of generic time-series models, which are employed for generating forecasts at global level $(\tilde{l}(t))$ and group level $(\tilde{l}_{g_k}(t))$. The *change detector module* constantly monitors the prediction accuracy and takes decisions on model change based on user-defined criteria. The adaptive prediction component provides a series of statistics that can be used for estimating model accuracy or for defining criteria for model selection or model change. The predictions generated by this component are forwarded to the adaptive data distribution component.
- The *adaptive data distribution* component allows to employ elastic data distribution strategies for content servers based on two modules: data grouping and data placement. The *data grouping module* dynamically clusters storage objects into logical groups, which are the unit of placement and replication on the content servers. The *data placement module* allows to implement adaptive data placement strategies based on the information provided by the adaptive prediction component.
- The *system sizing* module dynamically scales the system by turning servers on and off based on the

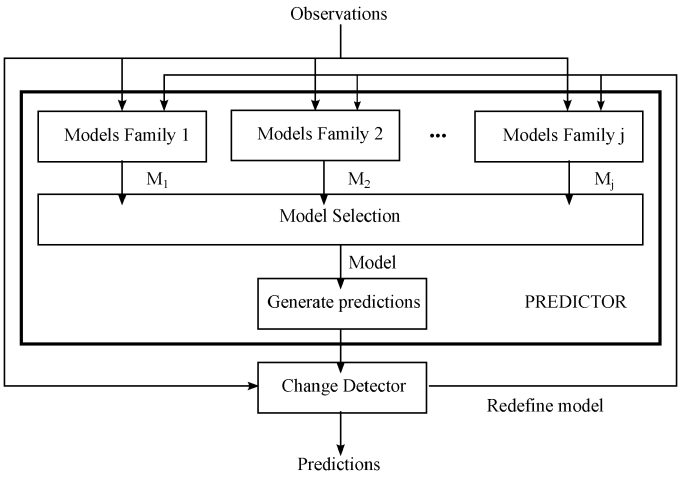

Fig. 2. Internal structure of the adaptive prediction component.

information provided by the adaptive prediction module.

The remainder of this paper focuses on the three components of the CONDESA framework: adaptive system prediction (Section 3), system sizing (Section 4), and adaptive data distribution (Section 5).

3 Adaptive Prediction Component

The adaptive prediction modeling component uses historical workload information and prediction accuracy for controlling the system size, data grouping, and data placement strategies. The logic of this component is implemented in two interacting modules, as shown in Fig. 2: a predictor module and a change detector module. The predictor module automatically defines prediction models, selects among several candidates and generates predictions from these models as discussed in Section 3.1. The change detector module estimates the model accuracy and decides when to redefine a model as described in Section 3.2.

3.1 Prediction Module

The prediction module manages models for predicting the system workload based on access pattern history. The prediction module receives as input a series of past observations $(l_{t-k}, \ldots, l_{t-2}, l_{t-1})$ and generates a time window of h predictions $(\tilde{l}_t, \ldots, \tilde{l}_{t+h-1})$ and the fitting error. The next three sections discuss the definition of prediction models (Section 3.1.1), selection of prediction models (Section 3.1.2), and the hierarchical modeling approach employed in CONDESA (Section 3.1.3).

3.1.1 Model Definition

The definition of a model is done in two steps: model formulation and model estimation (fitting) [7]. The model formulation identifies the form of the internal equations used by the models, while the fitting stage estimates the model parameters by methods such as recursive least squares or maximum likelihood.

CONDESA library contains currently three time-series family models: ARIMA [7], Holt-Winters [8], and AR_z [9]. For each of them CONDESA relies on automatic model formulation and estimation tools available for R statistical software environment. Nevertheless, CONDESA prediction library can be extended, as the prediction module interface is generic: any uni-dimensional time series family model can be incorporated into the framework and evaluated together with other system components.

Our choice of time series models is motivated by the fact that they are natural and straightforward ways of representing and predicting variability in time. This intuition has been confirmed by our experience, which has shown the appropriateness of applying time series to model workload variability.

3.1.2 Model Selection

CONDESA allows to employ several concurrent prediction models and to select the best one based on various criteria, as shown in Fig. 2. First, a candidate model is chosen based on a family model specific selector. For instance, CONDESA uses for autoregressive models the AIC and BIC [7] criteria to select the model with the best combination of low fitting error and small number of parameters. Further, a model can be chosen from the selected candidate models based on two criteria: *MinFE* and *MinPE*. Minimum Fitting Error (MinFE) selects the model with the smallest fitting error, while Minimum Prediction Error (MinPE) selects the model with the smallest prediction error. The error for both MinPE and MinFE is calculated as a Root Mean Squared Error (RMSE).

3.1.3 Hierarchical Modeling

CONDESA allows building hierarchical prediction models at two levels corresponding to models for the global and local workloads. A global model predicts the total workload of the system, which helps to dynamically size the system. At a second level, a local model predicts the expected number of requests for a group of items. This second level of prediction permits to identify workload variations at a lower granularity and to control the dynamic assignment of groups to content servers. The decomposition of the global workload into local workloads is managed by the data grouping module, which is part of the adaptive data distribution component and is discussed in Section 5.1. The relationship between global and local model predictions, data placement and system sizing is further discussed in Sections 4 and 5.2.

3.2 Change Detector Module

As the workload patterns change, the prediction models can become inaccurate and have to be either re-fitted or reformulated. The change detection module monitors the model accuracy and uses change detection criteria for discovering when a model prediction becomes inaccurate. Detecting when to redefine a model is a difficult task as it implies to distinguish between permanent changes in the workload and temporal fluctuations. CONDESA allows to define custom criteria for change detection. Below, we illustrate this process by presenting two change detection criteria we have already implemented in CONDESA for global and local models.

A model redefinition method periodically checks the accuracy of the predictions and decides based on change

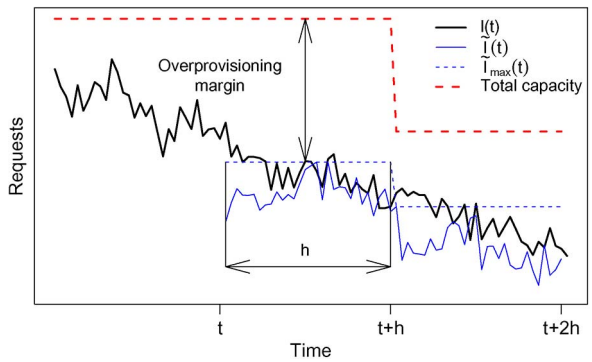

Fig. 3. System sizing for a prediction horizon h.

detection criteria when the models have to be redefined. The details of the change detection algorithms are provided in the supplementary material in Appendix B available online.

4 System Sizing Module

The system sizing module elastically scales the system by turning servers up and down. Fig. 3 shows the different variables involved in the system sizing process. First, at time t, the global workload model generates h predictions. In order to reduce oscillations in the system size evaluation, we take $\tilde{l}_{\max}(t, t+h-1) = \max(\tilde{l}(t), \ldots, \tilde{l}(t+h-1))$ as the reference value for system sizing for the next time window h. If we assume that all content servers have the same maximum capacity of c requests per unit of time ($c = c_s$ for all s) and that the desired utilization ratio is thr_u, the number of servers for the next prediction window h is given by:

$$\tilde{n}(t) = \ldots = \tilde{n}(t+h-1) = \left\lceil \frac{\tilde{l}_{max}(t, t+h-1)}{thr_u \cdot c} \right\rceil.$$

When the prediction is not possible or fails, the system sizing module allows to define reactive sizing policies. For example, if the observed load of a server $(l_s(t))$ is larger than a user configurable threshold, the system sizing module employs a reactive mechanism that allocates a new server on-demand. For instance, CONDESA can simply enforce the Amazon auto-scaling policy [10] by allocating a new server when the server load $l_s(t)$ exceeds a thr_u percentage of the server capacity c_s.

5 Adaptive Data Distribution

The adaptive data distribution component is in charge of dynamically adapting the data distribution based on discrete information provided from the adaptive prediction component. This component consists of two modules: the data grouping module and the data placement module.

5.1 Data Grouping Module

The data grouping module maps storage objects (data items) to placement groups. A placement group is a unit of data placement, i.e. it is not further divided in smaller logical units when stored on a server or on a disk (although it can be physically divided on several blocks when physically stored on a disk). CONDESA allows to associate prediction models with placement groups, i.e. our approach attempts to predict the workload at granularity of a group in order to detect local variations of workload and react accordingly. In order to make this approach efficient, groups have to be large enough to reduce the modeling overhead and small enough to avoid that popularity variations of individual items remain undetected.

CONDESA supports currently two data grouping policies: affinity-based grouping and random. Proposed in one of our previous works [3] affinity-based grouping builds placement groups based on the probability of accessing related content within a time window. This probability is computed based on the historical access patterns. Another grouping policy supported by CONDESA is on-demand random assignment of storage objects to placement groups. In the future we plan to implement and study further grouping strategies such as popularity-based regrouping [11] or entropy-based regrouping [12].

The placement groups are dynamically created though the data grouping strategies and can be used for in-memory or on-disk storage. Each placement group is assigned to one or several content servers as discussed in the next subsection.

CONDESA separates the mechanism of data grouping from the policy. This allows users to define and evaluate custom data grouping policies.

TABLE 1
Example of a Dispatching Table with 3 Placement Groups and 3 Servers. Each Group Is Replicated over a Number of Servers Based on Probability Intervals

Placement group	Content servers	Dispatching probability intervals
1	s_1	0 - 0.25
	s_2	0.25 - 0.5
	s_3	0.5 - 1
2	s_1	0 - 0.5
	s_2	0.5 - 1
3	s_2	0 - 1

5.2 Data Placement Module

The data placement module is in charge of managing the dynamic distribution of placement groups over content servers. This distribution is controlled through a data structure called dispatching table. The dispatching table maps placement groups onto content servers and is used for redirecting user requests to content servers. The details of the change dispatching table algorithm are provided in the supplementary material in Appendix C available online. Table 1 shows an example of a dispatching table. The table consists of three columns: placement groups, servers, and dispatching probabilities. Each group is associated with a set of servers with a certain dispatching probability. Each server is associated a probability interval, which controls the amount of requests redirected to each server at placement group granularity. To determine where to redirect a request, the dispatcher generates a uniform random number between 0 and 1 and identifies the server in charge of the group based on the probability distribution interval which contains that number. For example,

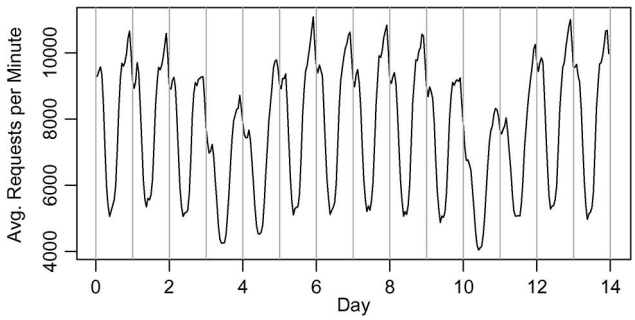

Fig. 4. Daily workload for the first two weeks.

assuming a request for an item belonging to group 1 using Table 1, the dispatcher generates a random number e.g., 0.48. As this number satisfies $p(s_1) < 0.48 \leq p(s_2)$, then the request is redirected to server s_2.

The dispatching table construction is based on assigning groups to servers and can be modeled as a bin-packing problem [13]: the groups g_j with sizes $\tilde{l}_{g_j}(t)$ are to be assigned to content servers $s_1, s_2, \ldots, s_n$, which are bins with capacities c_s.

The dispatching table plays a central role in the system dynamics, as it controls data and request distribution. The table is updated every time a new set of predictions is generated in order to assure an up-to-date view of the system. Additionally, the table has to be reactively recomputed every time the observed load exceeds the server capacity due to an unexpected event which causes a prediction failure.

6 Data Set

In Section 8 we demonstrate how CONDESA can be used for evaluating the design space of adaptive data placement policies. The evaluation is based on a set of traces collected from Wikipedia servers by Urdaneta *et al.* [14]. The whole data set contains 10 percent of all the requests directed to Wikipedia proxy caches from September 19th 2007 to January 2nd 2008 accounting for 20.6 billion requests. Each request in the trace contains a unique identifier, a time stamp, and the URL of the request. From this trace we selected only the requests to articles in the English Wikipedia during the first two weeks of trace. After filtering and cleaning the original trace, these two weeks account for 145 million requests to about 6 million data items.

Fig. 4 shows the number of requests per minute for the first two weeks. The workload is clearly periodic and similar to the one shown in [14]. There is a weekly pattern according to which the workload diminishes in weekends and increases during the week. During a single day, the number of requests doubles and decreases again. A similar daily pattern has been also observed in other systems [15], [16].

Unexpected workload variations are known to be common in the vast majority of web server infrastructures. However, the Wikipedia trace from Fig. 4 contains relatively regular seasonal variations, which can be relatively easy predicted by seasonal time series models. In order to be able to evaluate the power of adaptivity of CONDESA approach, we have extended the trace with real spike patterns extracted from other systems. We have considered two types of spikes: volume spikes and data spikes. A volume spike significantly modifies the global workload, while a data spike changes the popularity of small number of data items, while not necessarily impacting the total volume of requests.

We synthetically modify the Wikipedia dataset to include volume and data spikes following the methodology proposed by Bodík *et al.* [12] for generation of spikes for stateful Web services. We selected three significant spikes based on real traces: the requests served by Wikipedia after Michael Jackson's death (w_1), the server demand variations during the World Cup 1998 [17] (w_2), and a peak demand to Ebates.com servers [18] (w_3). Fig. 5 displays the resulting global workloads. The shadowed area represents the duration of the spike. For w_1 there is no significant change

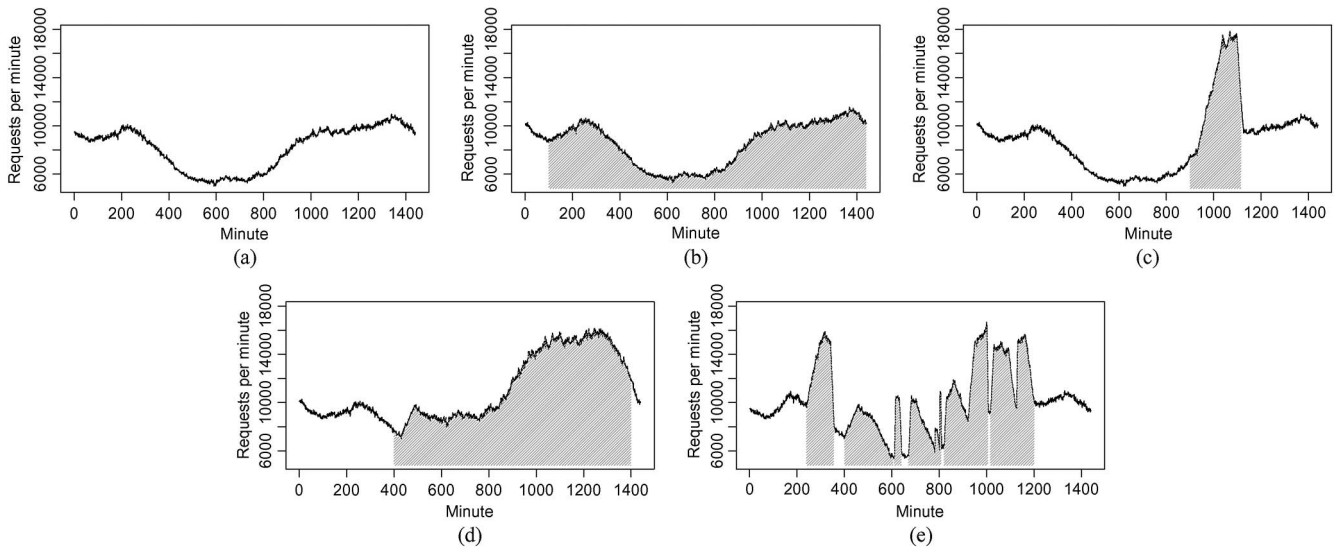

Fig. 5. Original workload and synthetically generated spike scenarios. Shadowed area indicates the duration and magnitude of the spike. (a) Original workload. (b) w_1. (c) w_2. (d) w_3. (e) w_4.

TABLE 2
Evaluation Parameters

Parameter	Value
Prediction models	Holt Winters (HW) ARIMA AR_z
Model selection criteria	MinPE MinFE
Change detection methods	GlobalCD LocalCD
h	30
η_G	0.5
ϵ_L	200
System sizing	Predictive Predictive+Adaptive
Data grouping	Random
Data placement	Uniform (*DP*) Predictive+Adaptive (*DP+P+A*)
d_s	512, 1024 and 2048 MBytes
thr_u	75%, 80%, 85%, 90%, 95%
c_s	2000

in the magnitude of the spike, only the popularity distribution of items varies. The scenario w_2 is characterized by a fast growth and fast decay. The spike w_3 is characterized by a steady growth followed by a moderately fast decay. In our experiments we employ the first three days of trace inserting the mentioned spikes in the third day. The three versions w_1, w_2 and w_3 account for 25.1, 25.5, and 28.6 million requests respectively. Finally, w_4 is a synthetic trace constructed by aggregating the original workload and ten spikes with random magnitude and duration.

7 CONDESA Metrics

This section presents a categories of metrics metrics used by CONDESA: system sizing metrics, server utilization metrics, and content locality metrics.

System sizing metrics. System sizing metrics estimate the efficiency of server provisioning based on the optimal number of servers. For a given server capacity c_s, the optimal number of provisioned servers can be calculated by:

$$n_{opt}(t) = \left\lceil \frac{l(t)}{c_s} \right\rceil$$

where $l(t)$ is the total number of requests received by the system at time t. We define the relative provisioning error $\eta_p(t)$ as:

$$\eta_p(t) = \frac{\tilde{n}(t) - n_{opt}(t)}{n_{opt}(t)}$$

where $\tilde{n}(t)$ is the number of provisioned servers at time t. Based on all calculated values of the relative provisioning errors in an time interval between t_1 and t_2, CONDESA calculates the overprovisioning rate $r_o(t_1, t_2)$ as the mean of all positive relative provisioning errors and the underprovisioning rate $r_u(t_1, t_2)$ as the negated mean of all negative relative provisioning errors (i.e. both r_o and r_u have positive values).

Server utilization metrics. CONDESA calculates the utilization of a server s at time t, denoted $u_s(t)$, as the ratio between the number of served requests l_s and the server capacity c_s. Based on these values CONDESA calculates aggregation metrics such as the average server utilization between t_1 and t_2 denoted $\overline{u}(t_1, t_2)$. The servers load balance for that interval is estimated as the standard deviation of server utilization and denoted $\sigma_u(t_1, t_2)$.

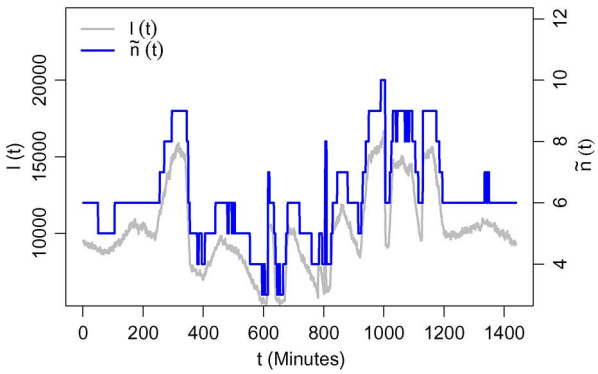

Fig. 6. Example of system size variation for w_4 workload, using $thr_u = 90\%$, random grouping, DP+P+A based on MinFE+A, and $d_s = 2048$ MB.

Locality metrics. Each server maintains an object cache with a configurable size d_s. To measure the content locality we evaluate the hit rate in each server. This hit rate for a period between t_1 and t_2 is denoted as $hit_s(t_1, t_2)$. The average for all the servers between t_1 and t_2 is denoted as $\overline{hit}(t_1, t_2)$.

8 Evaluation

This section demonstrates how CONDESA can be used for browsing the parameter space of the data distribution in an elastic server infrastructure. Due to space limitations we show only a small part of the analyses enabled by CONDESA. Table 2 overviews the input parameter values employed by the experiments discussed here. Further evaluations of modeling overhead, prediction accuracy, and change detection are provided in the supplementary material in Appendices D and E available online.

For brevity reasons we use the following naming conventions. Adding the suffix ''+P'' to a method name indicates the employment of prediction. Adding the suffix ''+A'' shows that a method employs model adaptivity based on change detection. For instance HW+A denotes the utilization of Holt-Winters with adaptivity enabled.

We evaluate a three-tier platform as the one showed in Fig. 1. For simplifying the analysis we make the following assumptions. The system has a variable number of dispatchers, which simply scale with the number of client requests. The client requests coming from the outside world are uniformly distributed over all dispatchers. The system has a variable number of content servers, whose elasticity is controlled by CONDESA. Finally, the storage system has a fixed size and a uniform service time.

This section is organized into four parts, which illustrate how CONDESA can be used for evaluating adaptive data placement strategies: system sizing (Section 8.1), server utilization and load balance (Section 8.2), data locality (Sections 8.3), over- and underprovisioning (Sections 8.4).

8.1 System Sizing

This section demonstrates how CONDESA can be used for evaluating the performance of system sizing.

Fig. 6 shows the global load $l(t)$ and the total number of provisioned servers $\bar{n}(t)$ for w_4 workload using $thr_u = 90\%$, random grouping, DP+P+A based on MinFE+A, and $d_s = 2048$ MB. We note that the provisioned servers adequately adapts to the workload volume and causes a small amount of oscillations.

Fig. 7 shows the saved machine time of various server provisioning approaches based on adaptive prediction, when compared with the traditional approach of statically provisioning a percentage over the maximum workload for w_1, w_2, w_3 and w_4. In this experiment, we statically provision 10 percent over the maximum workload i.e., $thr_u = 90\%$. For the provisioning based on adaptive prediction we use various prediction methods (ARIMA, ARIMA+A, AR_z, $\mathrm{AR}_z + A$, HW, HW+A, MinPE, MinPE+A, MinFE, and MinFE+A), $thr_u = 90\%$, random grouping, DP+P+A, and $d_s = 2048$ MB. The saved machine time is calculated by summing up the time individual machines are not provisioned due to predictions. The results indicate that the saved machine time is around 30 percent for w_1, w_3, and w_4 and 50 percent for w_2. The savings are larger for w_2 as this workload has a peak higher than the other workloads. In all cases the savings are significant and the adaptive prediction approaches can be used for turning off machines, and therefore, reduce the energy consumption.

8.2 Server Utilization and Load Balance

One of the most important objectives of any data placement technique is to maximize the server utilization, while perfectly balancing the load. CONDESA allows to estimate for a time interval (t_1, t_2) the average server utilization $\overline{u}(t_1, t_2)$ and the load balance $\sigma_u(t_1, t_2)$. In the optimal case the servers are 100 percent loaded and, thus, perfectly load balanced. However, this is impractical for two reasons. First, unexpected peaks will be delayed until additional resources will be available. Second, as the load of a server gets closer to 100 percent, the variance of the response time is known to increase [19]. Therefore, a small over provisioning of $1 - thr_u$ has the role of mitigating both problems.

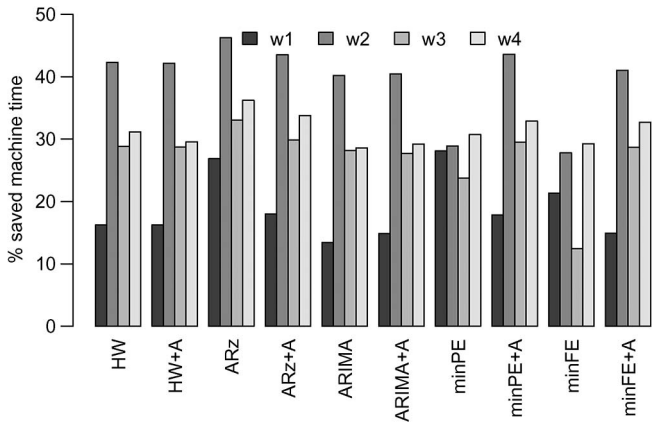

Fig. 7. Saved machine time over a static approach provisioning the maximum number of servers needed to serve the given workload. The example uses $thr_u = 90\%$, random grouping, DP+P+A, and $d_s = 2048$ MB.

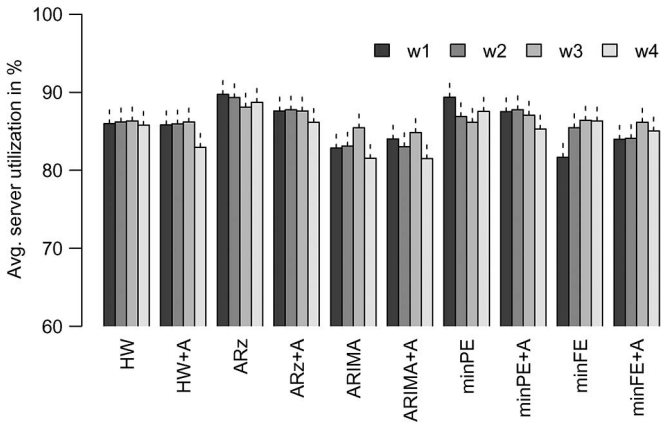

Fig. 8. Average server utilization $\overline{u}(t_1, t_2)$ (bars) and load balance $\sigma_u(t_1, t_2)$ (dashed lines) for w_1, w_2, w_3 and w_4, $thr_u = 90$, and DP+P+A data placement.

Fig. 8 shows how CONDESA allows to estimate $\overline{u}(t_1, t_2)$ and $\sigma_u(t_1, t_2)$ for an adaptive data placement policy, random grouping, $thr_u = 90\%$, three prediction models (HW, AR_z, ARIMA, MinPE, and MinFE), with and without adaptivity, and employing both global and local change detection.

The average server utilization does not exceed the threshold in any case. However, its closeness to the target threshold $thr_u = 90\%$ depends on the prediction method and on adaptivity. We observe that adaptivity (+A) improves the average server utilization in all cases. The load balance is stable: σ_u is approximately 2 percent for w_1, w_3 and w_4, and 3 percent for w_2.

8.3 Locality Evaluation

Another important indicator of the efficiency of a data placement policy is data locality. In this section we show how CONDESA allows to evaluate data locality of the content servers. Locality is measured as the average hit rate $\overline{hit}(t_1, t_2)$ over all the servers (individual server hit rates can also be retrieved). Fig. 9 shows a comparison of average hit rates between a data placement policy uniformly distributing the data over the content servers (DP) and a data placement policy employing predictive models and adaptivity based on MinFE+A (DP+P+A). Both policies scale

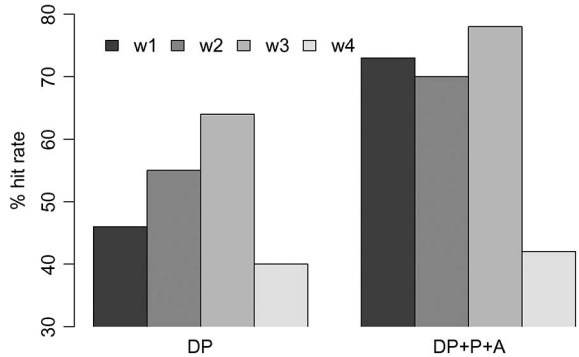

Fig. 9. Hit rate for w_1, w_2, w_3, and w_4 using $thr_u = 90\%$, random grouping, DP+P+A based on MinFE+A, and $d_s = 2048$ MB.

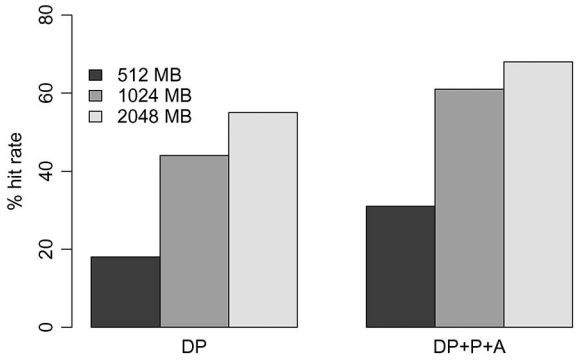

Fig. 10. Hit rate for w_2, using $thr_u = 90\%$, random grouping, DP+P+A based on MinFE+A, and various server cache sizes $d_s = 512$ MB, 1024 MB, and 2048 MB.

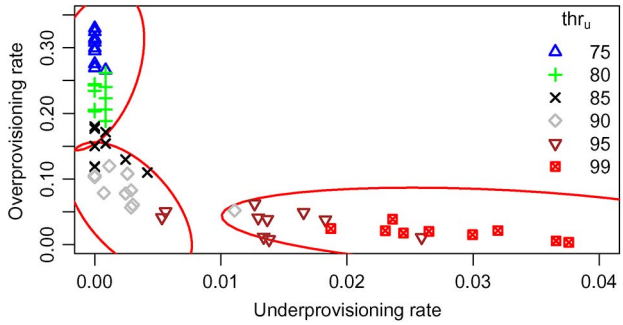

Fig. 11. Overprovisioning rate r_o and underprovisioning rate r_u for workload w_2 using random grouping, DP+P+A, several different values of thr_u (75 percent, 80 percent, 85 percent, 90 percent, 95 percent, and 99 percent), five prediction methods (ARIMA, AR_z, Holt Winters, MinPE, and MinFE), and enabled/disabled change detection for the prediction models.

based on the same system sizing policy described in Section 4 and are evaluated for w_1, w_2, w_3 and w_4 using $thr_u = 90\%$, random grouping, DP+P+A based on MinFE+A, and $d_s = 2048$ MB. The results show that DP+P+A obtains a significantly higher average hit rate than DP for all three workloads. In particular, this improvement is 25 percent for w_1 and 13 percent for w_2 and w_3.

In a second analysis, we are interested to see how the server cache size impacts the average hit rate for the two data placement policies from the previous experiment. We use the same parameters as in the previous experiment, except that we focus on w_2 workload and use three different cache sizes $d_s = 512$ MB, 1024 MB, and 2048 MB. Fig. 10 shows the results. As expected, the hit rate increases with the cache capacity for all methods. However, DP+P+A obtains around 15 percent improvement over DP for all three cache sizes.

DP+P+A uses prediction for estimating future load on servers, adaptation for changing prediction models when they become inaccurate, and proactive prediction-aware replication of content over servers as shown in Section 5.2. The higher hit rate for DP+P+A is explained by the fact that by accurately predicting the workload of the data placement groups and leveraging this prediction for pro-actively placing the data on servers results in higher temporal stability of content on servers. This temporal stability of content on servers means that a higher content locality can be obtained due to the fact that content is distributed/replicated based on workload prediction at group level.

8.4 Over- and Underprovisioning Evaluation

CONDESA facilitates the concomitant evaluation of over- and underprovisioning generated by an adaptive data placement policy. Fig. 11 shows the overprovisioning and underprovisioning rates for the workload w_2 using random grouping, DP+P+A, several different values of thr_u (75 percent, 80 percent, 85 percent, 90 percent, 95 percent, and 99 percent), three prediction methods (ARIMA, AR_z, and Holt Winters), two model selection criteria (MinPE and MinFE) and enabled/disabled change detection for the prediction models. For each thr_u value, there are ten points corresponding to the employed adaptive prediction method: ARIMA, ARIMA+A, AR_z, $AR_z + A$, HW, HW+A, MinPE, MinPE+A, MinFE, and MinFE+A. The x-axis of each point on the plot represents the underprovisioning rate r_u and the y-axis the overprovisioning rate r_o, respectively. An ideal method would have $r_u = 0$ and $r_o = 0$.

We observe in Fig. 11 three main clusters of points (a breakdown of the graph based on target utilization rates is provided in the supplementary material in Appendix F available online). First, the upper left cluster with r_o larger than 0.15 and r_u lower than 0.01 contains mostly points corresponding to target utilization rates thr_u of 75 percent and 80 percent. Second, the lower left cluster is the closest to the ideal under- and overprovisioning values and contains of r_o values between 0 and 0.15 and r_u values between 0 and 0.1 corresponding mostly to target utilization rates thr_u with values between 85 percent and 95 percent. Third, the lower right cluster contains points representing low r_o values and r_u values larger than 0.01, corresponding to target utilization rates thr_u with values larger than 95 percent. We notice that, as expected, as target system utilization gets closer to the 100 percent value, the underprovisioning rate increases, which has to be avoided. One would like to design a data placement strategy, whose corresponding r_o and r_u values fall in the lower left cluster. For our analyzed data placement policy, the best utilization threshold thr_u would be between 85 percent and 95 percent.

9 Related Work

Predictability is one of the most desired characteristics of a data center system, as it allows for optimal resource provisioning. In general, predicting high workload variations is a known difficult problem, and the safest approach is to substantially overprovision resources. The standard approach of capacity planning is to overprovision resources for the double of the expected peak load [20]. However, the advent of cloud computing has increased the need for proactive resource provisioning [21]. An important body of work has been dedicated to predictive methods of application workloads and dynamic resource allocations for data centers. Many prediction methods have been proposed based on queuing theory models [22], time series models [23], and machine learning techniques [24]. Several works employ autoregressive models for system size prediction [25], [26], [15], [24] and dynamically predicting CPU voltage/frequency in order to reduce power consumption [27].

Prediction techniques are useful for proactive resource provisioning to a limited extent. Unexpected events such as volume spikes and flash crowds require reactive resource provisioning for serving unexpected surges in workloads. Control theory has been recognized as a good match for addressing unpredictability and change in computing systems [28]. Even though many system designers still employ ad-hoc solutions of feedback control loops [29], there are an increasing number of works that consider the controllability as one of the design goals as discussed in the Appendix G of the supplementary material available online.

Workload decomposition has been used by several works for data and storage management. Some authors decompose the workload into a stable component and a variable component. For instance, Zhang *et al.* [30] describe a hybrid solution for private-public clouds that decompose the application workload into two components: a base workload and their spikes. The base workload is served from a private cloud, while spikes are served from instances launched in a public cloud. Some other works focus on decomposing the workload into partial workloads for groups of data rather than into stable and variable components. SCADS [31] is a control framework that monitors groups of items, and dynamically determines the number of storage servers and the most suitable replication strategy. Sastry and Crowcroft [32] propose to group popular user-generated content from an Internet workload onto few disks for allowing other disks to be placed in low energy states. CONDESA is a framework that allows to experiment with various workload decomposition techniques, while combining them with data grouping, data placement and system sizing strategies.

10 Conclusion

In this work we have presented CONDESA, a framework for exploring adaptive data distribution strategies for elastic server architectures. To the best of our knowledge CONDESA is the first platform that allows to systematically study the interplay between five data related strategies: workload prediction, adaptive control of data distribution and server provisioning, adaptive data grouping, adaptive data placement, and adaptive system sizing. We have demonstrated how CONDESA can be used for browsing the design space of adaptive data distribution policies. First, we have shown how prediction models can be compared in terms of overhead and accuracy. Second, we have evaluated the impact of change detection on prediction accuracy and how CONDESA can be used for choosing an adequate prediction horizon. Third, we demonstrated how the adaptive prediction can be used for sizing a server system. Fourth, we have shown how prediction models, change detection strategies and data placement policies can be combined and compared based on server utilization, load balance, data locality, over- and underprovisioning.

CONDESA is an evolving framework emerged from our previous work on adaptive data distribution of web server infrastructure. CONDESA evolution has opened up a considerable amount of possibilities to both extend the framework and apply it beyond its original domain. Currently, we are working on the integration of CONDESA with real infrastructures. Our objective is to provide an online tool for data distribution based on adaptive prediction. Further, CONDESA has been designed and implemented for facilitating the extension with novel modules for time-series prediction algorithms, model selection criteria, change detection criteria, data grouping and data placement. Finally, CONDESA can be used beyond the initial application domains, and we plan to apply it for designing and evaluating elastic data distribution policies for distributed file systems.

Acknowledgment

This research has been partially funded by the Spanish Ministry of Education under FPU program AP2007-03530 (Juan M. Tirado), and the Spanish Ministry of Science and Innovation under Grant TIN2010-16497 ''Input/Output techniques for distributed and high-performance computing environments.''

References

[1] A. Verma, R. Koller, L. Useche, and R. Rangaswami, ''Srcmap: Energy Proportional Storage Using Dynamic Consolidation,'' in *Proc. 8th USENIX Conf. File Storage Technol., ser. FAST'10*, Berkeley, CA, USA, 2010, p. 20, USENIX Association.

[2] L. Barroso and U. Holzle, ''The Case for Energy-Proportional Computing,'' *IEEE Comput.*, vol. 40, no. 12, pp. 33-37, Dec. 2007.

[3] J.M. Tirado, D. Higuero, F. Isaila, and J. Carretero, ''Predictive Data Grouping and Placement for Cloud-Based Elastic Server Infrastructures,'' in *Proc. 11th IEEE/ACM Int'l Symp. CCGrid*, May 2011, pp. 285-294.

[4] J.M. Tirado, D. Higuero, F. Isaila, and J. Carretero, ''Multi-Model Prediction for Enhancing Content Locality in Elastic Server Infrastructures,'' in *Proc. IEEE Int'l Conf. High Perform. Comput.*, 2011, pp. 1-9.

[5] J.M. Tirado, D. Higuero, F. Isaila, and J. Carretero, ''Reconciling Dynamic System Sizing and Content Locality Through Hierarchical Workload Forecasting,'' in *Proc. 18th IEEE ICPADS*, 2012, pp. 77-84.

[6] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, ''The State of the Art in Locally Distributed Web-Server Systems,'' *ACM Comput. Surveys*, vol. 34, no. 2, pp. 263-311, June 2002.

[7] R. Shumway and D. Stoffer, *Time Series Analysis and Its Applications,* 3rd ed. New York, NY, USA: Springer-Verlag, 2010.

[8] R.J. Hyndman and Y. Khandakar, ''Automatic Time Series Forecasting: The Forecast Package for R,'' *J. Statist. Softw.*, vol. 27, no. 3, pp. 1-22, July 2008.

[9] A.I. McLeod and Y. Zhang, ''Improved Subset Autoregression with R Package,'' *J. Statist. Softw.*, vol. 28, no. 2, pp. 1-28, Oct. 2008.

[10] Amazon, ''Amazon Auto-Scaling,'' 2012. [Online]. Available: http://aws.amazon.com/autoscaling/

[11] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, ''Scarlett: Coping with Skewed Content Popularity in Mapreduce Clusters,'' in *Proc. 6th Conf. Comput. Syst. EuroSys*, New York, NY, USA, 2011, pp. 287-300.

[12] P. Bodik, A. Fox, M.J. Franklin, M.I. Jordan, and D.A. Patterson, ''Characterizing, Modeling, and Generating Workload Spikes for Stateful Services,'' in *Proc. 1st ACM SoCC*, 2010, pp. 241-252.

[13] V. Vazirani, *Approximation Algorithms*. New York, NY, USA: Springer-Verlag, 2001.

[14] G. Urdaneta, G. Pierre, and M. van Steen, ''Wikipedia Workload Analysis for Decentralized Hosting,'' *Comput. Netw.*, vol. 53, no. 11, pp. 1830-1845, July 2009.

[15] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, ''Workload Analysis and Demand Prediction of Enterprise Data Center Applications,'' in *Proc. IEEE 10th IISWC*, Washington, DC, USA, 2007, pp. 171-180.

[16] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, ''Youtube Traffic Characterization: A View from the Edge,'' in *Proc. 7th ACM SIGCOMM IMC*, ACM, New York, NY, USA, 2007, pp. 15-28.

[17] M. Arlitt and T. Jin, ''A Workload Characterization Study of the 1998 World Cup Web Site,'' *IEEE Netw.*, vol. 14, no. 3, pp. 30-37, May/June 2000.

[18] P. Bodik, G. Friedman, L. Biewald, H. Levine, G. Candea, K. Patel, G. Tolle, J. Hui, A. Fox, M. Jordan, and D. Patterson, ''Combining Visualization and Statistical Analysis to Improve Operator Confidence and Efficiency for Failure Detection and Localization,'' in *Proc. 2nd ICAC*, 2005, pp. 89-100.
[19] P. Shivam, V. Marupadi, J.S. Chase, T. Subramaniam, and S. Babu, ''Cutting Corners: Workbench Automation for Server Benchmarking,'' in *Proc. USENIX Annu. Tech. Conf.*, 2008, pp. 241-254.
[20] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, ''Napsac: Design and Implementation of a Power-Proportional Web Cluster,'' in *Proc. 1st ACM SIGCOMM Workshop Green Netw. I*, New Delhi, India, Aug. 2010, pp. 15-22.
[21] M.E.A. Armbrust, ''A View of Cloud Computing,'' *Commun. ACM*, vol. 53, no. 4, pp. 50-58, Apr. 2010.
[22] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, ''An Analytical Model for Multi-Tier Internet Services and its Applications,'' *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291-302, June 2005.
[23] S. Casolari and M. Colajanni, ''Short-Term Prediction Models for Server Management in Internet-Based Contexts,'' *Decision Support Syst.*, vol. 48, no. 1, pp. 212-223, Dec. 2009.
[24] P. Bodik, R. Griffith, C. Sutton, A. Fox, M.I. Jordan, and D.A. Patterson, ''Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters,'' in *Proc. Workshop HotCloud*, 2009, pp. 1-16.
[25] A. Chandra, W. Gong, and P. Shenoy, ''Dynamic Resource Allocation for Shared Data Centers Using Online Measurements,'' in *Proc. SIGMETRICS*, New York, NY, USA, 2003, pp. 300-301.
[26] T.-H. Li, ''A Hierarchical Framework for Modeling and Forecasting Web Server Workload,'' *J. Amer. Statist. Assoc.*, vol. 100, no. 471, pp. 748-763, Sept. 2005.
[27] C. Santana, J. Leite, and D. Mossé, ''Power Management by Load Forecasting in Web Server Clusters,'' *Cluster Comput.*, vol. 14, no. 4, pp. 471-481, Dec. 2011.
[28] C. Karamanolis, M. Karlsson, and X. Zhu, ''Designing Controllable Computer Systems,'' in *Proc. 10th Conf. HOTOS*, Berkeley, CA, USA, 2005, vol. 10, p. 9, USENIX Association.
[29] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, ''What Does Control Theory Bring to Systems Research?'' SIGOPS Oper. Syst. Rev., vol. 43, no. 1, pp. 62-69, Jan. 2009.
[30] H. Zhang, G. Jiang, H. Chen, K. Yoshihira, and A. Saxena, ''Intelligent Workload Factoring for a Hybrid Cloud Computing Model,'' in *Proc. Int'l Workshop Cloud Serv.*, 2009, pp. 701-708.
[31] B. Trushkowsky, P. Bodík, A. Fox, M.J. Franklin, M.I. Jordan, and D.A. Patterson, ''The Scads Director: Scaling a Distributed Storage System under Stringent Performance Requirements,'' in *Proc. 9th USENIX Conf. FAST*, 2011, pp. 12-26.
[32] N. Sastry, A. Hylick, and J. Crowcroft, ''SpinThrift: Saving Energy in Viral Workloads,'' in *Proc. 2nd Int'l Conf. COMSNETS*, IEEE, 2010, pp. 1-6.

Juan M. Tirado received the PhD degree in computer science from the Carlos III University, Madrid, Spain, in 2013. He is a postdoctoral researcher at INRIA Rennes since 2013. His main research is on data analytics and how information can be employed to improve systems performance. His research includes cloud computing, and distributed and recommender systems.

Daniel Higuero received the PhD degree in computer science in 2013 from the Carlos III University, Madrid, Spain, where he was a teaching assistant from 2010 to 2013. His research interests are in distributed systems, efficient data distribution architectures, fast data transfers, and big data architectures.

Javier Garcia Blas received the PhD degree in computer science from University Carlos III, Madrid, Spain, in 2010. He has been a Teaching Assistant at the same university since 2005. He has also cooperated in several projects with researchers from various high performance research institutions including HLRS (funded by HPC-Europe program), DKRZ, and Argonne National Laboratory. He is currently involved in various topics including parallel I/O, parallel architectures, and energy-efficient storage solutions.

Florin Isaila received the MS degree from Rutgers University in 2000 and the PhD degree in computer science from University of Karlsruhe, Germany, in 2004. He is an Associate Professor in Computer Science at the University Carlos III, Madrid, Spain. He is a recipient of a Marie Curie International Outgoing Fellowship (2013-2016). He has been a Visiting Scholar at Argonne National Laboratory (2007-2008 and 2013-2015) and at Northwestern University (2006). His primary research interests include high-performance computing, storage, distributed systems, and data mining.

Jesus Carretero is a Full Professor of Computer Architecture and Technology at the Universidad Carlos III de Madrid, Spain, where he is responsible for that knowledge area since 2000. He is also Director of the Master in Administration and Management of Computer Systems, that he founded in 2004. He serves as a Technology Advisor in several companies. His major research are in parallel and distributed systems, real time systems and computing systems architecture.