

Applying Kalman Filters to Dynamic Resource Provisioning of Virtualized Server Applications

Evangelia Kalyvianaki
Computer Laboratory
University of Cambridge, UK
ek264@cl.cam.ac.uk

Themistoklis
Charalambous
Department of Engineering
University of Cambridge, UK
tc257@eng.cam.ac.uk

Steven Hand
Computer Laboratory
University of Cambridge, UK
smh22@cl.cam.ac.uk

ABSTRACT

Resource management in virtualized data centres is important and challenging, particularly when dealing with complex multi-tier server applications and fluctuating workloads. In this paper, we use control theory to build two controllers based on Kalman filters which monitor and vary CPU allocations across application tiers. Our approach (a) tracks utilisation patterns over noisy data, (b) considers the resource coupling among tiers and collectively allocates resources to them, and (c) adapts to workload conditions through an on-line parameter estimation mechanism. An initial experimental evaluation on a multi-tier server application shows that our controllers work effectively.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Measurement techniques, Modelling techniques.

General Terms

Measurement, Performance.

Keywords

Kalman Filter, Feedback Control, Resource Provisioning, Virtual Machines.

1. INTRODUCTION

System-level virtualization enables a single physical machine to host multiple *virtual machines* (VMs) which may run any application. In modern data centres, virtualization is commonly deployed to provide the abstraction of an agile set of resources. This can create VMs on demand and dynamically allocate resources to them.

A key prerequisite to high performance is setting these allocations as required to meet server application demands. Server applications, however, exhibit variable resource usage over time due to workload fluctuations; a simple static resource allocation scheme will fail to react to these changes. If under-provisioned, the application is not properly equipped to serve incoming requests; its performance will drop and Service Level Agreement (SLA) violations may occur. If over-provisioned, incoming requests are adequately served, but physical resources are under-utilised, thus preventing additional applications from being run. A dynamic resource allocation scheme is required to address these challenges.

Recently control theory has been used to perform dynamic allocation of CPU resources in virtualized data cen-

tres. CPU is an important resource for data centres. Server machines' reported under-utilisation has various consequences such as increase in power consumption. Therefore it is critical to use efficiently CPU resources. Dynamic techniques can be used to construct an efficient scheme that adjusts allocations as workload changes occur, without using extensive *a priori* knowledge of the workload patterns or the applications' internal structure. For example, Wang et al [9] present a nonlinear gain-adaptive integral controller that regulates the relative utilisation at a target value for single resource containers. The same controller in combination with other controllers has been studied in virtualized environments in order to: (a) allocate the CPU resources for co-located multi-tier applications [6]; (b) maintain the server response time within user-specified limits [12]; and (c) regulate the response time to a reference value with the aid of a performance model based on transaction mixes to better estimate the utilisation across tiers [8]. Finally, Liu et al [5] present an optimal controller that computes the resource allocations for multi-tier co-located virtualized applications, providing QoS response time differentiation in overload.

This paper presents a feedback control scheme that uses Kalman filters to control CPU resource allocation for multi-tier server applications deployed across multiple VMs. We formulate the allocation problem as a CPU utilisation tracking one, where a controller aims to maintain the CPU allocation at a certain limit above the utilisation. Tracking the utilisation is an intuitive approach to resource provisioning as each VM is allocated resources as needed.

Although others have also tried to regulate the relative CPU utilisation to a reference value [9, 6], the contribution of this paper is the integration of a very powerful filtering technique into a linear feedback allocation controller. Rather than using Kalman filters to estimate the parameters of an application performance model [11], we use Kalman filters as both as a tracking method *and* to build a feedback controller. The Kalman filter is particularly attractive since it is the optimal linear filtering technique when certain conditions hold and has good performance even when the conditions are relaxed.

This paper makes the following contributions: (a) a linear allocation controller for each VM based on tracking CPU usage; (b) an extended second controller that considers the resource coupling in multi-tier applications and enables faster allocations to saturated components; and (c) an on-line adaptation mechanism that modifies the filter parameters under different operating conditions without requiring any *a priori* knowledge.

sumed to be normally distributed:

$$p(z) \sim N(0, Q), \quad (4)$$

$$p(w) \sim N(0, R). \quad (5)$$

The measurement noise variance (R) might change with each time step or measurement. Also, the process noise variance (Q) might change in order to adjust to different dynamics. However, for the rest of this subsection they are assumed to be stationary during the filter operation. Later, we will present an approach which considers non-stationary noise.

Given that the equations (eq.) (2) and (3) describe the system dynamics, the required allocation for the next interval is computed based on tracking the utilisation and is a direct application of the Kalman filter theory. \tilde{a}_k is defined as the *a priori* estimation of the CPU allocation, that is the predicted estimation of the allocation for the interval k based on previous measurements. \hat{a}_k is the *a posteriori* estimation of the CPU allocation, that is the corrected estimation of the allocation based on measurements. The predicted *a priori* allocation for the next interval $k + 1$ is given by:

$$\tilde{a}_{k+1} = \hat{a}_k, \quad (6)$$

where the corrected *a posteriori* estimation over the previous interval is:

$$\hat{a}_k = \tilde{a}_k + K_k(u_k - c\tilde{a}_k). \quad (7)$$

At the beginning of the $k + 1$ interval the controller applies the *a posteriori* \hat{a}_k allocation. If the \hat{a}_k estimation exceeds the available physical resources, the controller allocates the maximum available. In the region where the allocation is saturated, the Kalman filter is basically inactive. Therefore, the filter is active only in the underloaded situation where the dynamics of the system are linear. The correction Kalman gain between the actual and the predicted measurements is:

$$K_k = c\tilde{P}_k(c^2\tilde{P}_k + R)^{-1}, \quad (8)$$

where \tilde{P}_k is the *a priori* estimation error variance and is calculated based on the *a posteriori* error variance \hat{P}_{k-1} :

$$\hat{P}_{k-1} = (1 - cK_{k-1})\tilde{P}_{k-1}, \quad (9)$$

$$\tilde{P}_k = \hat{P}_{k-1} + Q. \quad (10)$$

3.1.1 Modelling Variances

To obtain a good estimation of the allocation process noise variance Q it is enough to estimate the usage variance — since the allocation is considered to be proportional to the usage — and then evaluate it via the following formula:

$$var(a) \simeq var\left(\frac{u}{c}\right) = \frac{1}{c^2}var(u). \quad (11)$$

The usage process noise corresponds to the evolution of the usage signal in successive time frames. Estimating its variance is difficult, since the usage signal itself is an unknown signal and it does not correspond to any physical process well described by a mathematical law. The usage variance is calculated from measurements of the CPU utilisation. When the current BC controller is applied, the stationary process variance Q is computed off-line before the control process and remains the same throughout.

Finally, the measurement noise variance R corresponds to the confidence that the measured value is very close to the real one. Once more it is difficult to compute the *exact* amount of CPU usage. However, given the existence of relatively accurate measurement tools, a small value (e.g. $R = 1.0$ is used throughout the paper) can act as a good approximation of possible measurement errors.

3.2 Process Noise Covariance Controller

The Process Noise Covariance Controller (PNCC) further extends the BC controller by considering the resource coupling between multi-tier applications (MIMO controller in Figure 1, usages from all tiers are forwarded to all controllers). In multi-component servers, there is a correlation between the utilisation of the various tiers. Server applications are usually modelled with queues in tandem. If any of the tiers is inadequately provisioned, the overall server performance is affected. In fact, when workload fluctuations happen in resource provisioned server components, the saturation point can be moved from one component to another, causing prolonged poor server performance [7, 10].

To address this problem, the PNCC controller considers the coupling between the components. The allocation for each component is adjusted based on the errors of the current component in addition to the errors caused in the other components, as explained below. If n is the number of application components, then the PNCC Kalman filter equations for stationary process and measurement noise take the form:

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \mathbf{Z}_k, \quad (12)$$

$$\mathbf{U}_k = \mathbf{C}\mathbf{A}_k + \mathbf{W}_k, \quad (13)$$

$$\hat{\mathbf{A}}_k = \tilde{\mathbf{A}}_k + \mathbf{K}_k(\mathbf{U}_k - \mathbf{C}\tilde{\mathbf{A}}_k), \quad (14)$$

$$\mathbf{K}_k = \mathbf{C}\tilde{\mathbf{P}}_k(\mathbf{C}\tilde{\mathbf{P}}_k\mathbf{C}^T + \mathbf{R})^{-1}, \quad (15)$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{C}\mathbf{K}_k)\tilde{\mathbf{P}}_k, \quad (16)$$

$$\tilde{\mathbf{A}}_{k+1} = \hat{\mathbf{A}}_k, \quad (17)$$

$$\tilde{\mathbf{P}}_{k+1} = \hat{\mathbf{P}}_k + \mathbf{Q}, \quad (18)$$

where $\mathbf{A}_k \in \mathbb{R}^{n \times 1}$ and $\mathbf{U}_k \in \mathbb{R}^{n \times 1}$ are the allocation and usage vectors respectively and each row corresponds to a component; $\mathbf{C} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the target value c for each component along the diagonal; $\tilde{\mathbf{P}}_k \in \mathbb{R}^{n \times n}$ and $\hat{\mathbf{P}}_k \in \mathbb{R}^{n \times n}$ are the *a priori* and *a posteriori* error covariance matrices; $\mathbf{K}_k \in \mathbb{R}^{n \times n}$ is the Kalman gain matrix and $\mathbf{R} \in \mathbb{R}^{n \times n}$ and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ are the measurement and process noise matrices respectively. For matrices \mathbf{Q} and \mathbf{R} the diagonal elements correspond to the process and measurement noise for each component. The non-diagonal elements of the matrix \mathbf{Q} correspond to the process noise covariance between different components. Similarly, the non-diagonal elements of the \mathbf{K}_k matrix correspond to the gains between different components and are computed based on their covariances (eq. (15), (16), and (18)). For example, in a 3-tier application, the *a posteriori* $\hat{\mathbf{A}}_k(1)$ estimation of the allocation of the first component at time k is the result of the *a priori* estimation $\tilde{\mathbf{A}}_k(1)$ of the allocation plus the corrections from all components' innovations, given by:

$$\begin{aligned} \hat{\mathbf{A}}_k(1) = & \tilde{\mathbf{A}}_k(1) + \mathbf{K}_k(1, 1)(\mathbf{U}_k(1) - \mathbf{C}(1, 1)\tilde{\mathbf{A}}_k(1)) \\ & + \mathbf{K}_k(1, 2)(\mathbf{U}_k(2) - \mathbf{C}(2, 2)\tilde{\mathbf{A}}_k(2)) \\ & + \mathbf{K}_k(1, 3)(\mathbf{U}_k(3) - \mathbf{C}(3, 3)\tilde{\mathbf{A}}_k(3)). \end{aligned}$$

