# Automatic Request Categorization in Internet Services

Abhishek Sharma[*†], Ranjita Bhagwan[†], Monojit Choudhury[†], Leana Golubchik[*], Ramesh Govindan[*] and Geoffrey M. Voelker[§]

[†]Microsoft Research, India
[*]University of Southern California
[§]University of California, San Diego

## ABSTRACT

Modeling system performance and workload characteristics has become essential for efficiently provisioning Internet services and for accurately predicting future resource requirements on anticipated workloads. The accuracy of these models benefits substantially by differentiating among categories of requests based on their resource usage characteristics. However, categorizing requests and their resource demands often requires significantly more monitoring infrastructure. In this paper, we describe a method to automatically differentiate and categorize requests without requiring sophisticated monitoring techniques. Using machine learning, our method requires only aggregate measures such as total number of requests and the total CPU and network demands, and does not assume prior knowledge of request categories or their individual resource demands. We explore the feasibility of our method on the .Net PetShop 4.0 benchmark application, and show that it works well while being lightweight, generic, and easily deployable.

## 1. INTRODUCTION

As Internet and enterprise hosting services have evolved from single-host platforms to large data centers, the tasks of resource provisioning and capacity planning have similarly grown both in importance and difficulty. Performance modeling is a natural approach for accomplishing such provisioning and planning tasks in multi-tier cluster-based server systems. It forms the basis for many commercial tools [1,19], and contemporary models incorporate sophisticated techniques including detailed resource profiling of tiered services [18], long-term forecasting of workloads in enterprise systems [7], comprehensive queueing models of multiple service tiers [20], and the modeling of nonstationarity in workload mixes [17].

An important aspect of performance modeling is characterizing application workloads and systems to extract parameters, such as request rates and CPU service times, that serve as inputs to the models. Frequently, workloads are modeled as a single class and aggregate workload and system parameters are used as inputs. Application workloads, however, often have a variety of user interactions that affect resource demands. For example, user requests in e-commerce applications typically fall into various categories such as browsing and shopping. These workload categories can have different resource demands; for example, shopping requires more CPU than browsing. Moreover, the relative mix of those categories in the overall workload changes over time [7, 17]. As a result, models that explicitly incorporate a mix of workload categories and differentiate between them can more accurately perform resource provisioning and capacity planning for modern data centers [17].

Modeling workloads with category mixes, however, presents the challenge of parameterizing the different categories as inputs to performance models. Scalar (i.e., single class) models might require just the overall request rate $\lambda$ as an input, a straightforward measure to obtain from host performance counters or server logs. However, models with multiple request categories require request rates $\lambda_i$ for each category. Per-category parameters to be either extracted or inferred from the workload. Obtaining them would then require monitoring and logging infrastructure that both differentiates among categories as well as correlates requests across the multiple tiers of service infrastructure. Such instrumentation is certainly feasible to engineer, but often requires a deeper understanding of both the workload and the service architecture, as well as a substantially more complex monitoring infrastructure [2, 3].

This paper explores the problem of inferring workload categories without the need for sophisticated monitoring infrastructure. We present an automatic and generic request categorization method that uses only coarse-grained measurements of system resources, such as overall request rate, total CPU, and network usage as input. Using such measurements over multiple time windows, we apply a machine-learning based computational technique called independent component analysis (ICA) to differentiate and categorize requests and their resource demands in an offered workload. These per-category parameters can then be directly used as inputs in Internet service performance models.

A learning approach to workload categorization has a number of advantages. First, it greatly simplifies the profiling task when modeling workloads with multiple categories. Second, it treats both the workload and system as a black box, inferring request categories and resource usage without requiring detailed instrumentation. Finally, it can profile workloads at different granularities of categories. In addition to session-level categories such as browsing and shopping, it applies equally well to categorizing workloads in terms of content type (e.g., HTML, images, scripts) and request operation (e.g., GET and POST operations). Using ICA has constraints, though. It requires that certain statistical properties hold in the workload, and it limits the number of

[*†]Abhishek Sharma was an intern at Microsoft Research, India.

[†]Geoffrey M. Voelker was a visiting researcher at Microsoft Research, India from the University of California, San Diego.

.

categories that can be inferred to the number of measured aggregate resources. With our benchmark experiments, we show that workload categorization provides promising results despite these constraints.

This paper makes two contributions. First, we describe how the problem of request categorization maps to the framework that ICA uses. We also verify that our problem domain satisfies assumptions that the use of ICA requires. Second, we evaluate our model using the PetShop 4.0 [16] benchmark on a Windows Vista platform and show that, using ICA, we can efficiently derive the desired request categorization and resource usages. The percentage error for our request categorization estimates (relative to the ground truth) and our resource usage estimates (relative to an alternate, more heavy-duty technique) are within 4–17%.

The rest of the paper is organized as follows. Section 2 defines the problem and describes the model that we use to represent the resource usage in the different components of an Internet service. Section 3 describes how we map our problem to the ICA framework, and how the assumptions that ICA requires hold true for our data. We describe our experimental results in Section 4. In Section 5 we discuss the applicability of our request categorization technique, identify several promising directions for improving our technique, and discuss the ramifications of using our technique in large-scale Internet services. Section 6 describes related work. We conclude in Section 7.

## 2. MODEL

In this section we develop a mathematical model representing the problem of request categorization. Let there be $m$ basic *request categories* ($RC_1$ to $RC_m$) in an application workload. A request category represents a class of requests that have a similar resource consumption pattern. Let there be $n$ resources ($RS_1$ to $RS_n$) associated with the Internet service under consideration. Resources could be quantities such as CPU usage and network usage on web servers, database servers, and other service components.

We make a *linearity assumption* that states that there is a positive constant $a_{ij}$ such that the amount of resource $RS_i$ required for processing $x_j$ requests of category $RC_j$ is given by $a_{ij}x_j+c_{ij}$, where the constant $c_{ij}$ captures any non-linear components. In general, we can model the resource usage pattern for a given time window as a set of linear equations as shown below.

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{im}x_m + c_i = r_i \quad (1)$$

For example, say the index $i = 2$ gives us the equation for the web server's CPU usage. The total CPU usage on the web server is the sum of the individual CPU usages of each request made to the web server. Since there are $m$ different request categories, and since each request of category $k$ uses $a_{2,k}$ amount of CPU on the web server, the total CPU usage for the web server is a sum of the $m$ product terms shown on the left hand side of Eq. 1, plus the constant value. The constant value $c_i = \sum_{j=1}^{m} c_{ij}$.

Eq. 1 can be written in matrix form as

$$\mathbf{Ax} + \mathbf{c} = \mathbf{r} \quad (2)$$

where $\mathbf{A} = \{a_{ij}\}$ is the matrix of unit resource consumption for each request category, $\mathbf{x} = \{x_j\}$ is the vector of number of requests of a particular category, $\mathbf{c} = \{c_i\}$ is the vector of constant terms and $\mathbf{r} = \{r_i\}$ is the vector of the aggregate resource usage. In Section 4.2, we show that the above model does capture the CPU usage of a benchmark e-commerce application workload.

We further assume that the total number of requests, $\sum_{j=1}^{m} x_j$, is known to us; we address how this number can be derived in Section 4. We model this as the *first* equation of our framework of Eq. 1, or the first row of $\mathbf{A}$. So, for $i = 1$, all $a_{1,j}$ coefficients are set to 1 and the constant term $c_1$ is set to 0, so that $r_1$ is the total number of requests.

Note that when we gather a large number of samples over a period of time, $\mathbf{A}$ and $\mathbf{c}$ remain fixed, while $\mathbf{x}$ and $\mathbf{r}$ vary. Therefore, for a set of $T$ samples we can define a system of linear equations as follows.

$$\mathbf{AX} + \mathbf{C} = \mathbf{R} \quad (3)$$

with $\mathbf{X} = [\mathbf{x}^1\ \mathbf{x}^2 \dots \mathbf{x}^T]$ and $\mathbf{R} = [\mathbf{r}^1\ \mathbf{r}^2 \dots \mathbf{r}^T]$, where $\mathbf{x}^t$ and $\mathbf{r}^t$ represent the request arrivals for each category and the resource usage for the $t^{th}$ sample. The matrix $\mathbf{C}$ consists of $T$ repetitions of the column vector $\mathbf{c}$.

We model the (measurement) noise in our system as an additive zero-mean Gaussian. We incorporate the additive noise term into Eq. 3 as

$$\mathbf{AX} + \tilde{\mathbf{C}} = \mathbf{R} \quad (4)$$

where $\tilde{\mathbf{C}}_\mathbf{i} = \mathbf{C}_\mathbf{i} + \varepsilon$, with $\varepsilon$ being the additive noise. Geometrically, Eq. 4 says that if we plot each column of $\mathbf{R}$ as a point in an $n$-dimensional space, then the $T$ points corresponding to the $T$ columns of $\mathbf{R}$ should (approximately) lie on an $m$-dimensional hyperplane.

## 3. ANALYSIS

Eq. 4 describes the underlying linear model for resource usage in an Internet service. From the aggregate analysis of the resources consumed, we know the entries of $\mathbf{R}$. However, $\mathbf{A}$, $\tilde{\mathbf{C}}$ and $\mathbf{X}$ are unknown and neither do we know the number of categories, $m$, in the system. Therefore, the system seems to be extremely underspecified. Nevertheless, we can estimate $m$, $\mathbf{A}$ and $\mathbf{X}$ just from $\mathbf{R}$ using a machine learning technique called independent component analysis (ICA).

ICA is a gradient descent based optimization technique for solving the generic problem of "blind source separation" where the sources, $\mathbf{X}$, and the mixing matrix, $\mathbf{A}$, are unknown, but aggregate observations, $\mathbf{R}$, are known (see Ch. 7 of [11] for an accessible introduction). ICA can be applied to solve the matrix factorization problem as stated in Eq. 4 only if (a) the columns of $\mathbf{X}$ are statistically independent, and (b) the distribution of the columns of $\mathbf{X}$ is non-Gaussian.

Given that we know the entries of the $\mathbf{R}$ matrix, we can directly apply ICA to compute $\mathbf{A}$, eliminate $\tilde{\mathbf{C}}$, and thereby compute $\mathbf{X}$. However, before we do so, we must ensure that the basic assumptions of ICA hold for our data set and, secondly, we know the value of $m$ — the number of request categories.

*Non-Gaussianity of Request Arrivals*: The aggregate per second request arrivals to a web service can be modeled as samples drawn according to a geometric Poisson distribution (also known as Pòlya-Aeppli distribution) [12]. A geometric Poisson distribution for aggregate request arrivals results from modeling the client session arrivals as Poisson and the number of requests generated by each client session as a geometric random variable [12,22]. Since the distribution of the aggregate per second request arrivals is non-Gaussian, the

distribution of the aggregate request arrivals for each category (i.e., the distribution of the columns of $\mathbf{X}$) will also be non-Gaussian. For example, if the proportion of requests of each category in the workload is fixed, then the request rate for each category is a fixed fraction of the aggregate request rate.

Even though the aggregate per second request arrival is non-Gaussian, if we sample over large time intervals, the aggregate request arrivals can be approximately Gaussian. We can model the aggregate request arrivals over a sampling interval (larger than 1 second) as the sum of identically distributed random variables where each random variable models the aggregate request arrivals per second (i.e., follows a geometric Poisson distribution). Thus, the aggregate request arrivals for a large sampling interval can be Gaussian due to the central limit theorem. In our experiments (Section 4), we choose a small sampling interval equal to 10 minutes. The choice of sampling interval is an important parameter for our technique and is further discussed in Section 5.1.

*Independence assumption*: The assumption that the arrivals for different request categories (i.e., the columns of $\mathbf{X}$) are statistically independent may not always hold. In practice, ICA methods (e.g., the FastICA package that we use in Section 4), look for components that are as independent as possible. Techniques such as *whitening* the observed resource consumption measurements ($\mathbf{R}$) and randomizing the order of samples in the input to ICA helped reduce the error in our ICA based estimates for an e-commerce benchmark (Section 4.3). Whitening involves applying a linear transformation to a vector such that the resulting vector has uncorrelated components with their variances equal to unity. The FastICA package performs whitening by default. We further discuss the applicability of ICA based techniques to real world data in Section 5.4.

The problem of determining the number of request categories, $m$, boils down to finding out the *rank* of the $\mathbf{R}$ matrix. This can be understood as follows. If the data has been generated by $m$ request categories, but there are $n$ ($> m$) resources, then as stated earlier the $\mathbf{r}$ vectors will lie on an $m$-dimensional hyperplane in the $n$-dimensional space. This implies that all the samples of $\mathbf{r}$ can be expressed as a linear combination of $m$ basis vectors. Therefore, the rank of $\mathbf{R}$ should be $m$. Note that since the rank of $\mathbf{R}$ can at most be $n$, we cannot identify more request categories than the number of resource types we have.

Although there are several techniques for estimating the rank of a matrix, in practice we run ICA assuming $m = n$ to estimate $\mathbf{A}$ and $\mathbf{X}$. If $m < n$, then the entries of $n - m$ rows of $\mathbf{X}$ turns out to be significantly smaller (even negative) than the rest, which helps us throw away those request categories leaving only the $m$ valid ones.

The $\mathbf{A}$ matrix estimated by ICA is unique up to the scaling of the columns and their permutations. However, recall that the first row of the matrix models the total number of requests and, therefore, we require that $a_{1,j} = 1$ for all $j$. To ensure this, we can normalize the output of ICA. Suppose the matrix returned by ICA is $\mathbf{A}'$. To obtain $\mathbf{A}$, we perform the normalization step by dividing the entries of the $j^{th}$ column of $\mathbf{A}'$ by $a_{1,j}$. Also, since the different permutations of $\mathbf{A}$ correspond to different namings of the request categories, it is sufficient to know any one of the possible permutations.

# 4. EXPERIMENTAL VALIDATION

In this section, we validate the linearity assumption (Section 2) and demonstrate the feasibility of using our ICA based method for workload characterization using .NET PetShop 4.0 [16], a benchmark e-commerce application. We show that the linear model holds for CPU usage of the web and database servers. We also show that our method automatically categorizes requests, and determines good estimates of the number of requests and the resource requirements for each category.

## 4.1 Experimental methodology

We ran the ASP.NET PetShop 4.0 benchmark in two tiers: a front-end IIS 6.0 web server and a back-end SQL Server 2005 database. Both machines ran Windows Vista as the host operating system. A third machine running Visual Studio Team Suite (VSTS) [21] emulates multiple client sessions. Each client session generated either a browsing or a shopping workload session. Hence, in our experiments with session level categorization, the actual number of request categories is 2. A browsing session emulated a client merely browsing through various items on sale, while in a shopping session the client bought exactly one item after initial browsing. The average duration of shopping and browsing sessions was 30 and 45 seconds respectively. The different HTTP-level transactions within a session were generated using the VSTS record/replay utility: the recorder recorded a sample browsing session and a sample shopping session, and then replayed these sessions while varying user think time between transactions. The user think time was normally distributed with a mean of 3 seconds.

We collected 270 data points (matrix $\mathbf{R}$), each one at the end of a 10 minute-long experiment run. Hence, the value of $T$ is 270. Each data point consisted of 3 resource measurements: the aggregate CPU usage of the web server, the aggregate CPU usage of the database server, and the total number of client sessions serviced. Therefore, $n$ is set to 3. We measured the CPU usage using the `pstat` utility, available with the Microsoft Platform SDK for Windows Server 2003, and obtained the total number of client sessions serviced from the VSTS test run logs. In practice, we expect to use existing heuristics [9] to determine the total number of sessions using only the front-end web server logs. Before collecting the data points, we verified that a 10 minute experiment window was sufficient to obtain reproducible results. The number of emulated clients for each experiment run was randomly chosen between 10 and 200.

The objective of this experiment was twofold. First, we wanted to ensure that the linearity assumption holds for CPU usage by our benchmark workload, even though previous work has shown this to hold in large-scale Internet services [18, 23]. Second, we wanted to evaluate ICA as a method of automatic request categorization. That is, given $\mathbf{R}$ as described above, we wanted to validate whether our method automatically detects that the number of request categories is indeed 2 ($m$), that it accurately estimates the number of requests in each category in the different time slots ($\mathbf{X}$), and that it determines the per-category CPU usage on the web server and the database ($\mathbf{A}$).

We obtain the "ground truth" values for the number of client sessions serviced for the browsing and shopping sessions during each experiment (say $\tilde{\mathbf{X}}$) using the VSTS logs. We use these values to validate the linearity assumption

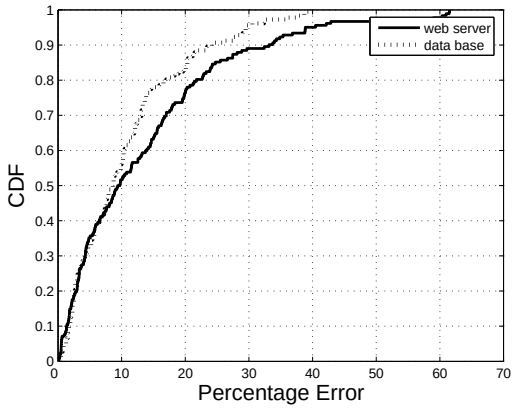


**Figure 1: IIS and SQL server CPU usage**

(Section 4.2), and as a baseline for comparing with the results inferred by ICA (Section 4.3).

## 4.2 Linearity validation

An F-test (a standard method for estimating data linearity) on the web server and the database server data, using $\mathbf{R}$ and $\tilde{\mathbf{X}}$, showed that the hypothesis that the data is linear can be rejected with a probability of 0. This clearly indicates the linear trend in the aggregate CPU usage on the web server and the database server.

We now show how much deviation there is from the perfect linear representation by comparing the measured values of resource usage to estimates that we obtain from the linear model. Figure 1 shows the cumulative distribution of the percentage error between the measured CPU usage values for the web server and the database server, and their estimates obtained using our linear model for resource consumption (Eq. 4).

To obtain the estimates for CPU usage from our linear model, first, we used the measured values for CPU usage ($\mathbf{R}$) and the ground truth number of browsing and shopping sessions ($\tilde{\mathbf{X}}$) to estimate the matrix $\mathbf{A}$ and the vector $\tilde{\mathbf{C}}$ using linear least-squares regression. Then, we use the estimates for $\mathbf{A}$ and $\tilde{\mathbf{C}}$, and the ground truth number of arrivals, $\tilde{\mathbf{X}}$, to obtained the estimates for CPU usage $\tilde{\mathbf{R}}$ using Eq. 4. The percentage error between the CPU usage estimate based on our linearity assumption, $\tilde{\mathbf{R}}$, and the measured values, $\mathbf{R}$, for a data point $i$ is computed as $\frac{|\tilde{\mathbf{R}}(i)-\mathbf{R}(i)|}{\mathbf{R}(i)} \times 100$.

For the web server and the database server CPU usage, the average percentage error was 13% and 11%, respectively. Approximately 85% of the data points for the database server and 75% of the data points for the web server had a percentage error less than 20%. Thus, for the complex workload generated by our benchmark e-commerce application, our linear model can capture the CPU usage fairly accurately. Similar results are reported by Zhang et. al [23] for the TPC-W benchmark workload in a two-tier setup consisting of a web server and a database server. For least-squares regression-based estimates, in the best (worst) case, they report a relative error of 15% for 98% (87%) of the samples for the web server and a 20% relative error for 89% (79%) of the samples for the database server.

| Category | Resource | Regression | ICA | Error |
|---|---|---|---|---|
| Shopping | IIS CPU | 66.87 | 69.69 | 4.2% |
| | SQL CPU | 13.27 | 14.86 | 12.0% |
| Browsing | IIS CPU | 44.44 | 47.58 | 7.1% |
| | SQL CPU | 3.27 | 2.76 | 15.6% |
| Category 3 | IIS CPU | - | -2053.26 | - |
| | SQL CPU | - | -23.81 | - |

**Table 1: Per-request CPU demand estimates for the PetShop benchmark (in milliseconds).**

## 4.3 Request categorization

As stated in Section 3, with three resources ($n = 3$) we can group the requests into at most three different categories. Lacking any *a priori* information, we start our ICA computation assuming that the number of categories is the same as the number of resources, i.e., $m = n = 3$. We used the FastICA [5] implementation of ICA to determine 3 sets of CPU usage coefficients for the web server and the database server for these request categories. We discuss how we infer the correct number of categories using ICA estimates later in this section.

*Resource consumption estimates.* Table 1 summarizes the estimated CPU demands of a single request of each category (entries of matrix $\mathbf{A}$) using FastICA and linear regression. The column labeled "Error" shows the relative difference between FastICA and linear regression estimates.

We make this comparison for the following three reasons. First, we lack the "ground truth" value for resource usage. Measuring the "ground truth" value for matrix $\mathbf{A}$ would require tracking the resource usage for each individual request within a session as it is serviced at the various tiers (web server and database server in our case) within the system. While this might be feasible using a monitoring tool like Magpie [2], it requires a thorough instrumentation of all the system components and may not scale to large numbers of simultaneous client sessions (the evaluation in [2] considered only 10 simultaneous clients). We plan to explore the feasibility of obtaining ground truth values for resource consumption using instrumentation in the future.

Second, as shown in Section 4.2, our linear model captures the aggregate CPU usage for the web server and database with small error. Hence, we expect the linear regression estimates of $\mathbf{A}$ to be fairly accurate and, thus, a natural yardstick for comparison in the absence of the ground truth values. Note that linear regression requires a fine-grained knowledge of the workload: the number of sessions of each category for each sample. Measuring the number of sessions of each category (required for linear regression) is not possible without the knowledge of the request categories present in a workload and, for an unknown workload, will require an instrumentation-based tool like Magpie [2]. We could use linear regression for our workload only because we knew the request categories and recorded the number of sessions for each category ($\tilde{\mathbf{X}}$) in our client emulator.

Third, comparing our matrix $\mathbf{A}$ estimates against linear regression provides a comparison between our method and an alternate technique in the literature. Linear regression is used in [23] to estimate the CPU usage for different (transaction level) request categories in the TPC-W benchmark workload.

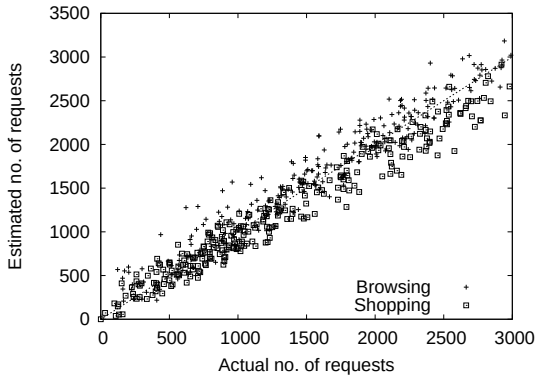


**Figure 2: PetShop: Request estimate comparison**

For the two request categories, i.e., shopping and browsing, the ICA estimates are quite close to the linear regression estimates. Our PetShop 4.0 workload consisted of two categories, browsing and shopping; but the ICA based method estimates 3 different categories. Additionally, the per-request CPU usage estimates for Category 3 are negative. We conjecture that FastICA models the noise in our system using the third category. This conjecture is further supported by the fact that 34% of the estimates for the number of sessions of Category 3 are also negative and that on an average the number of sessions in Category 3 was found to be only 0.6% of the total number of sessions.

*Number of sessions for each category.* We compare the FastICA estimates for the number of browsing and shopping sessions against the ground truth values collected from VSTS. Figure 2 shows a scatter plot with the ground truth values plotted on the x-axis and the estimates on the y-axis. The estimates are close to the actual values: the average percentage error is 11% and 17% for browsing and shopping categories, respectively. We believe that these errors can be further reduced using techniques that we discuss in Section 5.1.

# 5. DISCUSSION

Our results show that a lightweight, low-overhead and generic technique such as ICA is very useful in characterizing and categorizing requests. Using only aggregate metrics such as total CPU usage over time, we can estimate the number of request categories and per-request resource usages for an e-commerce application.

Much work remains towards increasing the accuracy of this technique. In this section, we first describe some of these directions. We also provide a brief overview of another blind source separation technique called Canonical Correlation Analysis (CCA) that we plan to compare against ICA as part of our future work. Finally, we discuss several design issues that we expect will arise in designing a capacity planning tool that incorporates our request categorization technique, especially for large Internet-scale systems.

## 5.1 Accuracy improvement

We plan to explore the following issues to evaluate the effectiveness of our ICA-based technique for workload characterization more rigorously, and to further improve its accuracy.

1) *Sensitivity analysis.* In general, a larger data set (larger value of $\mathbf{T}$) results in more accurate estimates. However, the minimum number of data points required to achieve accurate estimates depends on the workload. In general, the noisier the data, the higher the required value of $\mathbf{T}$. Guidelines for selecting $\mathbf{T}$ for different workloads and applications are a subject of future work.

The width of the monitoring window after which we collect a data point is another important parameter for our method. In general, deciding the duration of the monitoring window for a workload is tricky: a smaller time window will lead to more variability in the measurements, which is favorable to our categorization technique; but too small a time window may fail to capture a substantial number or requests in their entirety. For example, when modeling at the session granularity, typical shopping and browsing sessions last at least a few minutes. A smaller time window will then fail to model the system accurately. Hence, the duration of the monitoring window should be "long enough" to capture the processing of requests/client sessions in their entirety, but it should be "short enough" to prevent the aggregate request arrival rates from following a Gaussian distribution (as discussed in Section 3). We plan to do more experiments to investigate the impact of sampling duration on our ICA based technique. In particular, we are interested in using aperiodic sampling where the duration of the monitoring time window is chosen uniformly at random, for example, between 10 minutes to 1 hour.

2) *Effect of Noise.* The presence of (measurement) noise makes the problem of estimating the matrix $\mathbf{A}$ more difficult (see Ch. 15 of [11]). In practice, the measurements can be pre-processed to reduce noise before performing ICA. Even though FastICA is known to be consistent in the presence of noise with certain characteristics, an appropriately designed pre-processing step to reduce noise can further improve the estimate for matrix $\mathbf{A}$. Another alternative would be to use techniques specifically designed for noisy ICA [10].

3) *Only partially blind.* We are also exploring a more informed method of performing automatic request categorization by augmenting our ICA-based technique with feature-based information from server logs. For example, if we assume that, in general, HTTP GET requests are less resource-intensive than HTTP POST requests, or that static content is less resource-intensive than dynamic content, then we can adopt a more informed approach to request categorization. We believe that such feature-based differentiation can provide useful hints to our factorization algorithm.

## 5.2 Workload characterization granularity

One limitation of the current technique is that the number of request categories that we can identify is limited by the resources whose usage we can measure. Heterogeneous components within the system can help us identify more request categories. For example, in a system with separate web, application logic and database servers, using aggregate CPU usage measurements we can partition the requests into up to three different categories. We can also monitor usage of other resources — for example, it is straightforward to incorporate network usage into our model. More generally, with $s$ different components (servers) and $k$ different resources in each, we can partition the requests into at most $s \times k$ different groups.

## 5.3 Alternatives to ICA

Principal Component Analysis (PCA) and Canonical Correlation Analysis (CCA) are two other data-driven techniques for Blind Source Separation (BSS). All the three techniques (PCA, ICA and CCA) are designed based on a common principle: they aim to find linear transformations of the original data such that the resulting components (source signals) are mutually *uncorrelated*. The uncorrelated constraint, follows from the assumption that the original sources are independent. However, uncorrelatedness is not enough to distinguish between the possibly infinite number of linear transformations producing uncorrelated components, and hence, additional constraints are required.

PCA tries to maximize the variance of the resulting components. ICA tries to find statistically independent components by minimizing gaussianity (based on measures such as kurtosis, negentropy, Chapter 8 [11]). The justification for minimizing gaussianity to search from independent components comes from the central limit theorem. According to the central limit theorem, sums of nongaussian random variables are closer to being gaussian than the original variables. Hence, a linear combination of the observed mixture (e.g., the aggregate resource usage measurements in our case) will be maximally nongaussian if it equals one of independent sources. CCA maximizes the autocorrelation of the resulting components. The autocorrelation constraint is based on the observation that several real world sources (e.g., fMRI source signals) are autocorrelated [6]. ICA and CCA have been known to outperform PCA in terms of accuracy and robustness [6, 11]. In practice, the observed mixture is preprocessed using PCA to reduce its dimension before using ICA or CCA. For a detailed comparison of the performance of the three methods on functional MRI data, refer to [6].

We plan to compare the results from CCA based workload characterization with our ICA results as part of our future work. The advantages of using a CCA based method over ICA are the following: a) CCA does not require the original source components to be nongaussian, b) CCA does not require the original sources components to be independent (uncorrelatedness is enough) and c) two different runs of CCA on the same data identify the same components but the components identified by ICA in two different runs are not necessarily the same. This is so because packages such as FastICA use random starting points and hence, their output is not deterministic.

## 5.4 Request Arrivals: Assumptions

The blind source separation techniques discussed in the previous section look for source signals that are mutually uncorrelated. In addition, ICA tries to find components that are as independent as possible. If the request arrivals are correlated, then estimating mutually uncorrelated and/or independent components might not yield satisfactory results. The question whether BSS techniques like ICA and CCA are useful for real world workloads is an empirical one. We plan to evaluate our BSS techniques using workload benchmarks such as TPC-W and RUBiS as well as real production system traces as part of future work. It is worth mentioning here that BSS for correlated sources is an active area of research within the signal processing community [15].

## 5.5 Linear model applicability

The usefulness of techniques like ICA and CCA for workload characterization depends on the validity of our linear model (equation 4). Our validation results in Section 4.2 and the results in [17, 23] show that assuming a linear model for resource consumption is not an oversimplification, and that it can estimate the resource consumption in real production systems as well as benchmark workloads with low error.

However, the linear model may not be applicable to all scenarios. As pointed out in [17], the linear model ignores interaction effects across transaction types. For example, checkout transactions might require more CPU time during heavy browsing if the browsing activity reduces the cache hit rates for the checkout transactions. The likelihood of these nonlinear interaction effects across request/transaction types is much higher in a heavily loaded or an overloaded system. Stewart et. al [17] specify that their linear model based technique is most effective when the system is not heavily loaded–peak CPU utilization is less than 70%.

Additionally, our linear model does not capture autocorrelation in the aggregate resource utilization. Mi et. al [14] show that, in spite of independent generation of requests of a certain category, the measured resource consumption can show significant autocorrelation; especially when the request arrival is bursty. Furthermore, for a closed-loop system, this autocorrelation in demand can propagate to all the tiers in the system. Experimental evaluation in [14] shows that the observed autocorrelation in resource consumption is significantly stronger when the resource utilization at the bottleneck resource is above 80%, i.e. the system is heavily loaded.

Based on the results in [14,17], we conjecture that the linear model is applicable to scenarios where the system is not heavily loaded (e.g., peak resource utilization is less than 70%). Cockcroft et al. recommended that peak resource utilization in Internet services should be below 70% [4]. Thus, the linear model is applicable to a wide range of scenarios present in real production systems. We plan to investigate the applicability of our linear model in more depth as part of future work. We also plan to characterize the impact of autocorrelation and interaction across request types on the errors in our BSS based estimates.

## 5.6 Data collection issues

Internet services today consist of multiple components, each of which performs a particular task and can consist of hundreds, if not thousands, of machines. The existence of more components helps our request categorization technique by allowing a more fine-grained differentiation of requests, as discussed in Section 3. But each component can have multiple machines performing similar functions either to provide fault-tolerance or load-balancing. Since our method associates resource usage with a component, we would need to aggregate resource usage measurements from all machines providing the services of the component and calculate an average. If different groups of machines in the component perform diverse tasks, our problem formulation considers each group as a separate resource and therefore will collect resource usage statistics from each group as an independent measure.

# 6. RELATED WORK

Performance modeling of multi-tier Internet services is an area of active research. In this section, we discuss some

of the work related to performance modeling and workload characterization.

The proposed techniques in published literature for workload characterization and performance modeling can conceptually be classified into two categories–*inference* based and *instrumentation* based. The key challenge for inference-based techniques is calibrating the parameters of an analytical model from aggregate statistics such as overall CPU utilization. Instrumentation-based techniques provide a more detailed characterization of the system as compared to the inference-based techniques, but require invasive middleware or kernel level instrumentation and an extensive monitoring and logging infrastructure.

## 6.1 Inference-based techniques

Inference based techniques typically construct an analytical model for the system and perform workload characterization to estimate the parameters of the analytical model. For example, multi-tier web applications have been modeled as a network of queues in [13, 20, 23]. The number of request categories, the arrival rate for each request category and the resource requirements (service times) for each request category are the key parameters needed to calibrate a queueing model. To estimate these parameters, inference based methods use aggregate resource utilization (for example, CPU usage) measurements and the information available in server logs.

Liu et al. assume that the number of request categories and their arrival rates at each queue/service tier is known [13]. They use measured server utilization and end-to-end request delays to estimate the service times (resource requirements) at each tier for each request category. The estimates for the service times are such that the weighted least square error between the measured and the estimated server utilization (using the queueing model) and end-to-end request delay is minimized. Minimizing the least square error requires solving a quadratic program and the authors present an efficient algorithm for it.

Zhang et al. assume a linear relationship between the aggregate CPU utilization at servers and the resource usage by each category [23]. This linear relationship is the same as our linear model (equation 4). Assuming knowledge of the number of request categories and the arrival rate for each category, the authors estimate the service times for each category using ordinary least squares regression. Zhang et al. use the TPC-W benchmark for experimental evaluation and treat each transaction (for example, add an item to a cart) as a separate request category.

Urgaonkar et al. develop a product-form queueing model of multi-tier Web services for estimating request response times [20]. Using high-level system and workload parameters, such as request rates and service times, the model can predict response times for various system configurations. Many of their results are in terms of a single overall workload category, but they also show how to extend their queueing model to incorporate different known request classes. The model for scalar workloads requires request rates and residence times at all tiers in the system. The model, extended with classes, needs these parameters for all categories.

Stewart et. al use an analytical model to estimate the request response time (end-to-end delay) based on the observed requests for each transaction types (the transaction mix) and the aggregate resource utilizations [17]. The authors use a linear model similar to equation 4 to model the aggregate resource utilization. In order to estimate the model parameters using benchmark data, the authors perform Least Absolute Residuals (LAR) regression. Results based on benchmark experiments and analysis of real production system data show that the approach works well when none of the resources are saturated.

Our blind source separation based approach to Internet services workload characterization is an inference based technique. However, unlike the related work described above, we do not assume that various request categories and their arrival rates are known. In the published literature, a common approach when dealing with benchmark experiments with multiple request categories is to assume that each transaction type represents a separate category [17, 23]. This approach is based on the assumption that the different transaction types have different resource demands (e.g., "checkout" is more CPU intensive than "browsing"). While this approach has been shown to yield acceptable performance (within 10-20% relative prediction error) for benchmarks such as TPC-W and RUBiS, it does not work for workloads involving transaction types with similar resource usage (refer to the evaluation with the StockOnline benchmark in [17]). In addition, identifying the transaction types for a web service workload often requires a deeper understanding of the workload and the system architecture, as well as significant experience with managing a real production system.

Using resource utilization measurements and information from server logs, BSS based techniques (ICA and CCA) can estimate both the number of request categories and their arrival rates, as well as the resource usage at each service tier by requests of each category, provided that the assumptions required for BSS are satisfied. These parameter estimates can then be used to calibrate the various inference models, thus eliminating the need to know number of request categories and their arrival rates *a priori*. Note that our ICA based approach requires the same amount of information as other inference based techniques, namely, aggregate resource utilization measurements and information from server logs.

An alternate approach to request categorization has been proposed by Goldszmidt et. al [8]. They define a feature vector consisting of request statistics (e.g., total number of instances for each URL, mean number of active instances, number of sessions), measured resource utilization and the response time for each request. All this information can easily be obtained from server logs. The authors use K-means clustering to categorize URLs according to the average system load while processing the URLs and the response time required to process the URLs. The work in [8] also defines a metric called the Effective e-Business Capacity (EEC). EEC is the number of e-Business jobs (e.g., client sessions) that a web service can service without violating a pre-determined service-level agreement (e.g., 90% of the requests generated by the client sessions should have a response time less than 1 second). Given the request categories, several machine-learning based techniques (e.g., Bayes' classifier and logistics regression classifier) are used to infer whether the system has enough EEC to satisfy the service-level agreement for a given session arrival rate. Like our BSS based technique, the inference models in [8] are automatically and dynamically built from measured data. However, unlike the queueing based inference models, the EEC based models in [8] cannot capture the impact of an individual component on

system performance (e.g., the impact of a CPU or memory upgrade).

## 6.2 Instrumentation-based techniques

It is possible to obtain a detailed workload characterization — request categories, arrival rates for each category and the per-category resource usage at each service tier — using a monitoring and logging infrastructure. Tools such as Magpie [2] and PinPoint [3] collect fine-grained information about each request including the various servers (e.g., web, application and database servers) visited by a request, and the resource usage at these servers during each visit. This fine-grained information can be used to cluster requests into different categories and to determine the per-category resource usage. While these tools have been implemented as research prototypes, the complexity of the monitoring infrastructure and the multi-tier service architecture, and the expertise required to deploy and use these tools, has hindered their adoption in production systems.

# 7. CONCLUSION

This paper explores the feasibility of using independent component analysis for Internet service application workload characterization. Using measurements of aggregate system resource usage, such as CPU, our method can infer various characteristics of the workload: the number of different request categories, the resource demands of each category, and the distribution of request arrivals across different categories. Inferring these category parameters has direct application to the performance modeling of tiered Internet application hosting systems. These models can benefit substantially from the increased fidelity of modeling workloads in separate categories. The advantage of our approach is that it does not require detailed instrumentation of the system to obtain category characteristics, just aggregate workload information that is straightforward to obtain. Our evaluation results using an e-commerce benchmark are promising and indicate that our method can potentially characterize fairly complex workloads with acceptable error.

# 8. ACKNOWLEDGEMENT

We thank our shepherd Cathy H. Xia, John Douceur and the anonymous referees for their constructive suggestions that improved the paper's presentation.

# 9. REFERENCES

[1] S. Bagchi, E. Hung, A. Iyengar, N. Vogl, and N. Wadia. Capacity planning tools for web and grid environments. In *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*, October 1996.

[2] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *Proceedings of the OSDI'04*, December 2004.

[3] M. Y. Chen, A. Accardi, E. Kiciman, A. Fox, D. Patterson, and E. Brewer. Path-Based Failure and Evolution Management. In *Proceedings of the NSDI'04*, March 2004.

[4] A. Cockcroft and B. Walker. *Capacity Planning for Internet Services*. Sun Press, 2001.

[5] The FastICA package for Matlab and R. `http://www.cis.hut.fi/projects/ica/fastica/`.

[6] O. Friman, M. Borga, P. Lundberg, and H. Knutsson. Exploratory fMRI Analysis by Autocorrelation Maximization. *NeuroImage*, 16(2):454–464, 2002.

[7] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *Proceedings of the IISWC'07*, September 2007.

[8] M. Goldszmidt, D. Palma, and B. Sabata. On the Quantification of e-Business Capacity. In *Proceedings of the Electronic Commerce*, 2001.

[9] X. Huang, F. Peng, A. An, and D. Schuurmans. Dynamic Web Log Session Identification With Statistical Language Models. *Journal of the American Society for Information Science and Technology*, 55(14):1290–1303, 2004.

[10] A. Hyvarinen. Gaussian moments for noisy independent component ananlysis. *IEEE Signal Processing Letters*, 6(6), June 1999.

[11] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley-Interscience, 2001.

[12] J. Judge. A Model for the Marginal Distribution of Aggregate Per Second HTTP Request Rate. In *Proceedings of the 10th IEEE Workshop on Local and Metropolitan Area Networks*, 1999.

[13] Z. Liu, L. Wynter, C. X. Xia, and F. Zhang. Performance inference of queueing models for IT systems using end-to-end measurements. *Performance Evaluation*, 63(2006):36–60.

[14] N. Mi, Q. Zhang, A. Riska, E. Smirni, and E. Riedel. Performance Impacts of Autocorrelated Flows in Multi-tiered Systems. In *Proceedings of the Performance'07*, October 2007.

[15] W. Naanaa and J.-M. Nuzillard. Blind source separation of positive and partially correlated data. *Signal Processing*, 85(9):1711–1722, 2005.

[16] Microsoft .NET Pet Shop 4.0. `http://msdn2.microsoft.com/`.

[17] C. Stewart, T. Kelly, and A. Zhang. Exploiting Nonstationarity for Performance Prediction. In *Proceedings of the EuroSys'07*, March 2007.

[18] C. Stewart and K. Shen. Performance Modeling and System Management for Multi-component Online Services. In *Proceedings of the NSDI'05*, May 2005.

[19] TeamQuest model: Capacity planning software with modeling. `http://www.teamquest.com/`.

[20] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, , and A. Tantawi. An Analytical Model for Multi-tier Internet Services and its Applications. In *Proceedings of the SIGMETRICS'05*, June 2005.

[21] Microsoft Visual Studio 2005 Team Suite. `http://msdn2.microsoft.com/`.

[22] K. H. Yeung and C. W. Szeto. On the Modeling of WWW Request Arrivals. In *Proceedings of the International Conference on Parallel Processing Workshops*, 1999.

[23] Q. Zhang, L. Cherkasova, and E. Smirni. A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications. In *Proceedings of the ICAC'07*, June 2007.