Predictive Modeling of Streaming Servers

Michele Covell, Sumit Roy, Beomjoo Seo
Hewlett-Packard Laboratories, Palo Alto CA

1 Introduction

In this paper, we describe our approach to deriving streaming-server saturation models from vector-labeled training data. If a streaming server is driven into saturation by accepting too many clients, the quality of service degrades across the sessions. The actual saturating load on a streaming server depends on the detailed characteristics of the client requests, whether Video-on-Demand (VoD) or live-cast, the relative popularity, and the bit and packet rates [1]. Previous work in streaming-server models has used carefully selected, low-dimensional measurements, such as client jitter and rebuffering counts [2], or server memory usage [3]. In contrast, we collect 30 distinct low-level measures and 210 nonlinear derivative measures each second. This provides us with robustness against outliers, without reducing sensitivity or responsiveness to changes in load. Since the measurement dimensionality is so high, our approach requires the modeling and learning framework described in this paper.

We take the approach of predictive classification (saturating/non-saturating) through hidden variable estimation, i.e. $\mathbf{s}$, the L-dimensional usage levels of critical resources on the server. We estimate these L resource-usage levels as affine combinations of $\mathbf{m}$, the 240-dimensional extended measurement vector, using *critical-resource models*, $\mathbf{R}_l$ $1 \le l \le L$:

$$\mathbf{s} = [\mathbf{R}_1 \cdots \mathbf{R}_L]^T \begin{bmatrix} 1 \\ \mathbf{m} \end{bmatrix}$$

For admission-control decisions, we increase the current resource-usage estimate ($\mathbf{s}$) by $\Delta\mathbf{s}$, the additional resource usage needed to support the extra clients. An N-dimensional vector, $\Delta\mathbf{c}$, counts the *increase* in the number of sessions, for each client category. For our work, $N = 8$ corresponds to the outer product of session type (VoD/live-cast), popularity (high/low), and bit-rate (high/low). We use $\mathbf{C}_n$, $1 \le n \le N$, a linear model on the resource usage by clients:

$$\Delta\mathbf{s} = [\mathbf{C}_1 \cdots \mathbf{C}_N]\, \Delta\mathbf{c}$$

By combining these two resource usage estimates $\mathbf{s} + \Delta\mathbf{s}$ and thresholding according to the desired server-loading/reliability trade-off, we provide admission-control decisions. We do not need to know the identities or the number of current sessions (an approach that would require open-ended bookkeeping), we just need the requested number of additions to the session load.

We describe a new mathematical framework for deriving the models needed in our approach. Some of the components of our framework are based on standard methods, like support-vector machines (SVMs), projection-onto-convex-sets (POCS), total least squares (TLS), and inequality-constrained quadratic solution. We show that a novel combination of these techniques is required to solve this modeling problem.

2 Predictive-Model Derivation

We take a labeled-training approach to determine $\mathbf{R}_l$, the resource models of the server, and $\mathbf{C}_n$, the client models giving the usage-profile of these resources. We use $\underline{\mathbf{m}}_t$ and $\underline{\mathbf{c}}_t$ to represent the training-set measurement and label samples that are below saturation and $\overline{\mathbf{m}}_t$ and $\overline{\mathbf{c}}_t$ to represent those that are above saturation. The 240-dimensional measurement vectors, $\underline{\mathbf{m}}_t$ and $\overline{\mathbf{m}}_t$, already include noise-variance equalization across measurement dimension and percentile filtering of the measurements over 60-second intervals (to reduce the effect of outlier measurements). The label vectors, $\underline{\mathbf{c}}_t$ and $\overline{\mathbf{c}}_t$, are the N-dimensional counts of requested sessions across client category.

Our first step is to estimate $\mathcal{F}_n$, a nominal saturating load for the n^{th} client category. This is done by finding the optimal convex saturated/unsaturated-decision boundary between $\{\underline{\mathbf{c}}_1, \ldots, \underline{\mathbf{c}}_{\underline{T}}\}$ and $\{\overline{\mathbf{c}}_1, \ldots, \overline{\mathbf{c}}_{\overline{T}}\}$,

under the constraint that this boundary touches the inscribing hypercube on all N client-category axes: then $\mathcal{F}_n$ are the dimensions of that inscribing hypercube. This could be done by decision-boundary learning techniques, such as SVMs. Since we only need $\mathcal{F}_n$, we take the approach of starting from calibration data on the N axes of our client-count sub-vector and estimate each $\mathcal{F}_n$ independently, to maximally separate the saturated and unsaturated data points on the n^{th} axis.

Once we have $\mathcal{F}_n$ for $1 \leq n \leq N$, we define $\mathbf{x}_t = \text{diag}\{\frac{1}{\mathcal{F}_1} \cdots \frac{1}{\mathcal{F}_N}\}\ \mathbf{c}_t$, an N-dimensional normalized client-load label, giving $\underline{\mathbf{x}}_t$ for the unsaturated training set and $\overline{\mathbf{x}}_t$ for the saturated one. Using $\mathbf{m}_t$ and $\mathbf{x}_t$, the models are developed by:

- estimating $\underline{\mathbf{s}}_t$ and $\overline{\mathbf{s}}_t$, the L-dimensional resource usage, as $[\mathbf{C}_1 \cdots \mathbf{C}_N]\,\mathbf{x}_t$, a linear function of the N-dimensional client-load vector, for $\mathbf{x}_t = \underline{\mathbf{x}}_t$ and $\mathbf{x}_t = \overline{\mathbf{x}}_t$, respectively.
- estimating $\underline{\mathbf{s}}_t$ as $[\mathbf{R}_1 \cdots \mathbf{R}_L]^T \left[1\ \underline{\mathbf{m}}_t^T\right]^T$, an affine function of the extended measurement vector.
- forcing $\max_l(\underline{\mathbf{s}}_t) = \underline{\mathbf{x}}_t$ so that one resource usage will reach 100% at the saturation boundary and no resource usage will be greater than 100% below the saturation boundary
- forcing $\max_l(\overline{\mathbf{s}}_t) \geq 1$ so that at least one resource usage is over 100% above the saturation boundary
- forcing $\min_l(\underline{\mathbf{s}}_t) \geq 0$ and $\min_l(\overline{\mathbf{s}}_t) \geq 0$ so that resource usage is never negative

$\mathbf{C}_{l,n}$ denotes the l^{th} dimension of the vector $\mathbf{C}_n$. We use $E(n) = \arg\max_l \mathbf{C}_{l,n}$ to map the client category n to l, the resource dimension that it uses more than other resources. We alternately solve our constraint sets, similar to POCS. We initialize by setting the number of resources, L, to the number of client types, N; the mapping function to $E(n) = n$, the identity function; the client model $\hat{\mathbf{C}}_n$ to a unit vector along the n^{th} dimension; and the threshold for client-model change C_δ to one (allowing maximum change). The first set of constraints creates a model of the resources from the measurement set, using weighted TLS under inequality constraints to minimize the residual over $\mathbf{R}_l$:

$$\mathbf{R}_l^T \begin{bmatrix} 1 & \cdots & 1 \\ \underline{\mathbf{m}}_1 & \cdots & \underline{\mathbf{m}}_{\underline{T}} \end{bmatrix} = [\mathbf{C}_{l,0} \cdots \mathbf{C}_{l,N}] \left[\underline{\mathbf{x}}_1 \cdots \underline{\mathbf{x}}_{\underline{T}}\right]$$

over $0 \leq \mathbf{C}_{l,n} \leq 1\ \forall l$; $\mathbf{C}_{E(n),n} = 1$; $|\mathbf{C}_{l,n} - \hat{\mathbf{C}}_{l,n}| \leq C_\delta$; and $||\mathbf{R}_l||_2 \leq R_{\max}^2$.

$R_{\max}$, the maximum norm allowed for the resource model vector $\mathbf{R}_l$, avoids over-training. $\hat{\mathbf{R}}_l$ minimizes the TLS residual matrix under these constraints. The next step estimates the resource usage, $\underline{\mathbf{s}}_t$, for each element in the below-saturation training set, using $\hat{\mathbf{R}}_l$ and $\underline{\mathbf{m}}_t$:

$$\underline{\mathbf{s}}_t^T = \left[1\ \ \underline{\mathbf{m}}_t^T\right] \left[\hat{\mathbf{R}}_1 \cdots \hat{\mathbf{R}}_L\right]$$

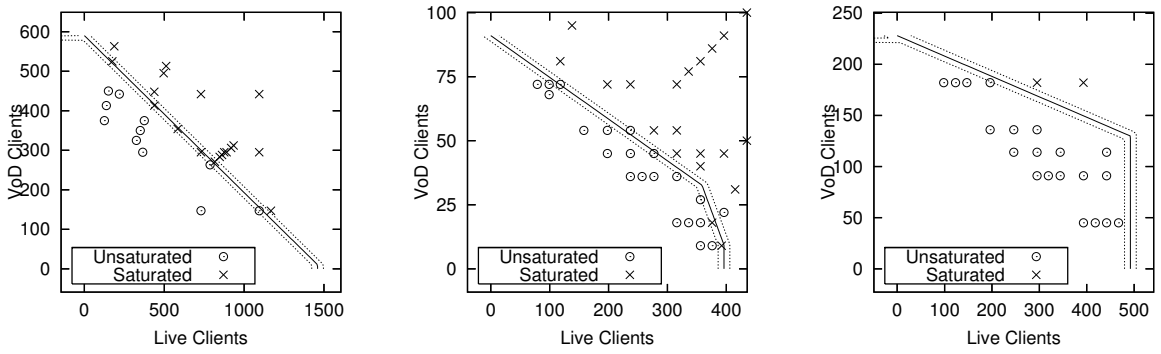
projected, element-wise onto the interval $[0\ \ ||\mathbf{x}_t||]$. Next, the client-usage models, $\hat{\mathbf{C}}_n$ are updated, again using TLS under inequality constraints on $\mathbf{C}_n$:

$$[\mathbf{C}_1 \cdots \mathbf{C}_N] \left[\underline{\mathbf{x}}_1 \cdots \underline{\mathbf{x}}_{\underline{T}}\right] = \left[\underline{\mathbf{s}}_1 \cdots \underline{\mathbf{s}}_{\underline{T}}\right]$$

over $0 \leq \mathbf{C}_{l,n} \leq 1\ \forall l$; $\mathbf{C}_{E(n),n} = 1$; and $\left|\left|[\mathbf{C}_1 \cdots \mathbf{C}_N] \left[\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_{\overline{T}}\right]\right|\right|_\infty \geq 1$.

We then reduce the value of C_δ by 25% and return to the $\mathbf{R}_l$-estimation step. Using this approach on our streaming-server training data, we see convergence within 10 iterations. If C_δ is not reduced periodically, we have seen undamped oscillations between the model pairs, due to our non-convex objective functions.

The models thus derived can be used without further reduction. Since there is no constraint on the orthogonality [4] or independence of the resource models [5], over-estimating the number of critical resources does not negatively impact the usage and saturation hypotheses. However, when we want to use the resource and client models to guide our selection of hardware subsystems, minimal representation of the critical resources maps them onto simpler combinations of distinct subsystems. This resource reduction is achieved



(a) popular, high-bitrate (b) unpopular, high-bitrate (c) unpopular, low-bitrate

Figure 1: **State-estimation surfaces dictated by the client-resource model for mixed VoD/live-cast.** The dotted lines show the model surfaces for 97% and 102% loading on the critical resource for the client mixture.

by an extra step of estimating the number of critical resources. We examine the resource-usage profile across all clients. If two resources "look similar enough" in terms of client usage of them, they are merged and the estimation process is repeated on this reduced resource set. We take pairs of resources, l_1 and l_2 for $l_1 < l_2$, and determine the correlation coefficients between their usage across all clients:

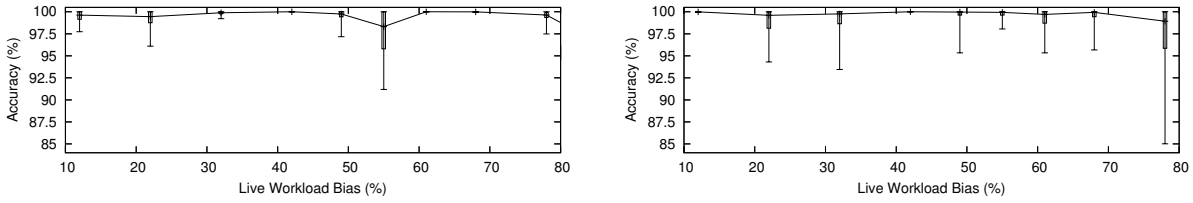
$$\rho_{l_1,l_2} = \frac{\left[\begin{array}{ccc} \mathbf{C}_{l_1,0} & \cdots & \mathbf{C}_{l_1,N} \end{array}\right] \left[\begin{array}{ccc} \mathbf{C}_{l_2,0} & \cdots & \mathbf{C}_{l_2,N} \end{array}\right]^T}{\sqrt{|||[\mathbf{C}_{l_1,0} \cdots \mathbf{C}_{l_1,N}]||_2 ||[\mathbf{C}_{l_2,0} \cdots \mathbf{C}_{l_2,N}]||_2}}$$

We then find the resource pair that has the maximum correlation coefficient. If that ρ_{l_1,l_2} is greater than some threshold (e.g., 95%), we merge the two resources. To do this, we remap $E(n) = l_2$ to $E(n) = l_1$ and we set $\hat{\mathbf{C}}_{l_1,n}$ to the average between $\hat{\mathbf{C}}_{l_1,n}$. We reset C_δ to one and then repeat the previously described resource and client models with $L - 1$ in place of L, using these values as our starting points.

3 Experimental Results

We evaluated the accuracy of our modeling work using the *Helix* server [6] loaded by mixing the 8 distinct client-session types. Experimental details are described in [1]. We train and test on disjoint loading conditions. This is stricter than just using disjoint data sets: each training vector lies on one of the 8 client-category axes, each testing vectors is off-axis. For each of our test (and training) vectors, we determined whether the server was overloaded by observing the quality of service at the clients. Figure 1 shows a scatter plot of the test vectors. We use the constraint equation: $|| \left[\hat{\mathbf{C}}_1 \cdots \hat{\hat{\mathbf{C}}}_N\right] \mathbf{x}||_\infty \leq 1$ to draw the convex surface which the client models predict as the boundary between under-saturated and over-saturated loads. The 97%- and 102%-saturation boundary surfaces show the sensitivity of the models.

Figure 1 shows good agreement between the predicted and observed behavior on mixed-client loads. Given the client-load vector, we can use $\hat{\mathbf{C}}_n$ to predict the saturation of the server under that load and can make admission-policy decisions based on that prediction. By using the actual client-load vectors to draw the boundary surfaces, Figure 1 side steps the estimation of the current server load. Figure 2 includes that current-state estimation as well as modified-load state prediction. We collapsed our results from eight dimensions down to one, for ease of visualization. The remaining dimension is the bias between VoD and live-cast. We use *box-and-whiskers* plotting to show the mean, standard deviation (vertical boxes) and extrema (vertical lines) of our results.



(a) State Estimation (b) State Prediction

Figure 2: **Server Load Estimation:** *State Estimation* accuracy is determined by whether $\mathbf{Rm}_t$ correctly predicts the under-/over-saturated status of the server. Ground-truth saturation was determined by running the server under the client load and observing whether or not it maintained the required quality of service. Similarly, *State Prediction* accuracy is determined by whether $\mathbf{R}\underline{\mathbf{m}}_{t_1} + \mathbf{C}(\mathbf{x}_{t_2} - \underline{\mathbf{x}}_{t_1})$ correctly predicts this bi-level value of the server saturation under the loading indicated by $\mathbf{x}_{t_2}$. The t_1 (starting-load) sample is always taken from the unsaturated data set. The t_2 (target-load) sample can be taken from either saturated or unsaturated data sets. The x-axis is the VoD/live-cast bias that was used to create the testing load.

4 Conclusions

In this paper, we described our approach to hidden-variable modeling and prediction. By estimating the current server load from low-level measurements, we allow this modeling approach to be applied to third-party software, without internal monitoring of the software status. By using statistical learning to derive the resource descriptions, we create models that are tailored to the given software operating on the given hardware configuration. Since this derivation does not require expert intuition, these well-tuned models of the software/hardware performance can be determined without expensive manual intervention. Furthermore, the models can be updated as more data is collected from QoS monitoring services.

We plan to examine the effects of model-adaptive reduction in the measurement vectors that are collected from the streaming server machine and from the probe clients. The coefficients in the resource models tend to be highly concentrated on a small number of measurement dimensions, with more than 95% of the model-vector norm derived from less than 10% of the measurement dimensions. By recursively omitting measurements during our calibration stage and retraining, we should be able to reduce the size of the measurement vector, while minimizing the impact of this reduction on the prediction performance. Folding this process into the calibration stage will allow the data reduction to be responsive to the specific streaming server software and hardware.

References

[1] M. Covell, B. J. Seo, S. Roy, M. Spasojevic, L. Kontothanassis, N. Bhatti, and R. Zimmermann, "Calibration and Prediction of Streaming-Server Performance," Tech. Rep. HPL-2004-206R1, Hewlett-Packard Laboratories, 2004.

[2] A. C. Dalal and E. Perry, "A new architecture for measuring and assessing streaming media quality," in *Passive and Active Measurement Workshop*, (La Jolla CA), 2003.

[3] L. Cherkasova, W. Tang, and A. Vahdat, "Mediaguard: a model-based framework for building qos-aware streaming media services," in *SPIE Conference on Multi-Media Computing and Networking*, 2005.

[4] M. Knop, J. Schopf, and P. Dinda, "Windows Performance Monitoring and Data Reduction using Watch-Tower," in *Proc. of the Workshop on Self-Healing, Adpative, and Self-Managed Systems*, (June), 2002.

[5] Y. Feng, V. Zarzoso, and A. K. Nandi, "Quality Monitoring of WDM Channels with Blind Signal Separation Methods," *Journal of Opitcal Networking*, 2004.

[6] RealNetworks Inc., "Helix Universal Server." http://realnetworks.com/products/media_delivery.html.