AUTONOMIC MIX-AWARE PROVISIONING FOR NON-STATIONARY DATA CENTER WORKLOADS

Derek Bender
PROBLEM

- Internet applications face spikes in varying types of workloads
  - Flash crowds
- SLA violations due to under-provisioning
- Simple server provisioning not enough
PURPOSE

• Provide sufficient resources for maximum application performance

• Handle changing applications demands
  • Volume
  • Mix – characteristics of the request
EXAMPLE I

- Web application with long and short requests
  - Short – 1ms
  - Long – 90ms
Example Session I

- Example Session I – 90 short and 10 long requests

- 100 req/s

- \(90 \text{ short} \cdot 1 \text{ ms} + 10 \text{ long} \cdot 90 \text{ ms} = 990 \text{ ms}\)

- 990 ms of server processing each second
EXAMPLE I SESSION B

• Example Session II – 10 short and 90 long requests
  • 100 req/s – remains unchanged
  • \(10 \text{short} \cdot 1\text{ms} + 90\text{long} \cdot 90\text{ms} = 3610\text{ms}\)
    • 3610ms of server processing each second
    • Rate has remained constant
    • Quadrupled processing demand!

Monday, March 7, 2011

A system that tracked aggregate request volumes would not notice any change in the request rate, and thus, would be unable to react to this large change in server load.
EXAMPLE II 75% LONG

• Request types unchanged – 90ms for long and 10ms for short
• Start with 50/50 split of long and short
• Request rate doubles from 100 req/s to 200 req/s
  • Results in unpredictable demand
• Mix of 150 long and 50 short
  • Doubled request rate, tripled server capacity
EXAMPLE II 75% SHORT

- Mix of 150 short and 50 long
  - Just a 2% increase in server capacity

- Naively doubling server capacity
  - under-provisioning
  - over-provisioning

\[
\frac{150 \times 1 + 50 \times 90}{50 \times 90 + 50 \times 1} = 1.02
\]
BENEFITS

- Successful implementation eliminates SLA violations
  - Due to under-provisioning
- Improved resource usage
  - Due to reduced over-provisioning

Better resource usage by reducing over-provisioning when compared to a baseline provisioning approach that only reacts to workload volume changes.
The technique employs the k-means clustering algorithm to automatically determine the workload mix and a queuing model to predict the server capacity for a given workload mix.
• Data Center Model

• Multi-tiered

• Cluster of commodity Servers

• High speed LAN – Gb/10Gb Ethernet
• Aggregator – monitors incoming workload statistics

• Mix determiner – characterizes application workload

• Predictor – estimates future workloads

• Actuator – dispatches and collects servers as needed
CHARACTERIZING WORKLOAD MIXES

• $k$-means clustering algorithm

• $n$ objects into $k$ partitions ($k < n$)
ALGORITHM STEP 1

- Determine optimal $k$ and partition unique request types into clusters
- All values of $k$ compute:
  - Coefficient of variation ($c = \sigma/\mu$) of intra- and inter-cluster distance
  - Ratio of these variances
  - Ratio of the coefficient of variation
  - The best $k$ value minimizes intra-cluster variance and maximizes inter-cluster variance

Optimize number of clusters ($k$)
Iterative approach
std. deviation over mean
ALGORITHM STEP II

• Adjust for heavy-tailed workloads

  • To prevent large workload averages, requests are broken up and handles by smaller cluster groups

  • Determined by a threshold value
ALGORITHM STEP III

• Computer Cluster Mean

• Average service demand for all requests on a cluster

• If a cluster has service times \( S = \{s_1, s_2, s_3, \ldots, s_k\} \) which appear with frequencies \( F = \{f_1, f_2, f_3, \ldots, f_k\} \), the cluster mean is given by:

\[
\frac{\sum_{i=1}^{k} s_i f_i}{\sum_{j=1}^{k} f_j}.
\]
ALGORITHM STEP IV

• Recomputation of cluster means
  • Adjust as workload mixes change
• Mix-oblivious scenario
  • Cluster mean changes frequently by large amounts
• With optimal cluster count
  • Cluster mean remains more stable

More stable even with workload changes
MIX-AWARE PROVISIONING

• Provisioning so that SLA is not violated

• Decisions by algorithm:
  • when to trigger provisioning
  • how much additional capacity to provision
PROVISIONING – WHEN?

• Periodic - once per 12 hrs.

• Triggered by thresholds:
  • Volume
  • Mix
• Three steps to determine amount to provision
  • Estimate $\lambda_i$ for each cluster to determine $\lambda_{total}$
  • Queuing model to predict capacity
    • $G/G/1$ queuing system
      • Kingman’s approximation for waiting time
**Input:** Let there be $k$ clusters at this tier, incoming volume of requests in each cluster $P = \{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_k\}$, the cluster mean of each cluster $D = \{d_1, d_2, d_3, \ldots, d_k\}$, the $95^{th}$ percentile response time threshold for this tier $y$, the variance of inter-arrival time $\sigma_a^2$, the variance of service time $\sigma_b^2$.

**Output:** Number of servers needed for this tier

\[
\lambda_{total} = \sum_{i=1}^{k} \lambda_i \\
\bar{x} = \frac{\sum_{i=1}^{k} \lambda_i d_i}{\sum_{i=1}^{k} \lambda_i} \\
\lambda_{per-server} = \left[ \bar{x} + \frac{\sigma_a^2 + \sigma_b^2}{2(y/3 - \bar{x})} \right]^{-1} \\
N = \left\lfloor \frac{\lambda_{total}}{\lambda_{per-server}} \right\rfloor \\
return N
\]
PROVISIONING – QUANTITY?

• Applying the new configuration

• Actuator adds or removes servers at each tier
Figure 3(a) shows $CV_{\text{inter}}$, $CV_{\text{intra}}$ and $\beta_{\text{cv}}$ with increasing values of $k$. Both $\beta_{\text{cv}}$ and $CV_{\text{intra}}$ reach their lowest values when $k = 5$ which is the optimal cluster size for this setup. Clustering performance degrades sharply with significantly higher values of $\beta_{\text{cv}}$ and $CV_{\text{intra}}$ with $k > 5$. $CV_{\text{inter}}$ remains stable with an increasing number of clusters.
Figure 3(b) shows how $\beta_{var}$ changes with an increasing number of clusters ($k$). $\beta_{var}$ reaches its smallest value for $k = 5$. Higher number of clusters gives higher $\beta_{var}$ values. These 2 figures indicate that the appropriate number of clusters is 5 for the database tier.
Error is estimate of service demand
Figure 3(c) shows the evolution of errors in the service demand estimates for both workload transitions. The x-axis represents the number of ordering mix sessions decreasing from 800 to 0. Depending on the experiment, the number of browsing and shopping sessions are increasing by 100 each time the ordering sessions decrease by 100. The maximum error observed was 5% for the transition from ordering to browsing. The error remains below 1% for the transition from ordering to shopping.
TEST CASE

Non-stationary workload with constant volume
TEST CASE

Monday, March 7, 2011

new products – 2.6ms
fast search – 5.5ms
slow search – 14.0ms
TEST CASE
TEST CASE

Provisioning logic invoked every 10 minutes
Mix aware anticipates capacity requirements sooner
Result: This experiment illustrates that when the volume of requests is constant the resource demand on a tier may still change because of a change in the workload mix. A mix-aware provisioning is therefore able to provision resources to account for these changes and avoid SLA violations that would be experienced with a provisioning scheme only looking at the volume of requests.
TEST CASE

Non-stationary workloads with a varying volume
TEST CASE
TEST CASE
TEST CASE
Even when the volume of requests is changing, a mix-aware provisioning policy refines traditional predictions based on volume changes by preventing both under-provisioning and over-provisioning.
EC2 TEST CASE

(a) workload mix for the EC2 case study
(b) mix-aware vs mix-unaware provisioning

EC2 TEST CASE
CONCLUSION

• *k*-means clustering algorithm

• Accurately captures workload mix changes

• Mix-aware dynamic provisioning > simple volume-based provisioning

• by eliminating SLA violations due to under-provisioning

• Better resource management by controlling over-provisioning