Transaction Mix Performance Models

Methods and Application to Performance Anomaly Detection

Terence Kelly kterence@hpl.hp.com Hewlett-Packard Labs, Palo Alto, CA

ABSTRACT

This poster describes a simple model of application-level performance as a function of workload. The model is intuitive, easy to apply, and requires little knowledge of the application. The model can be used for *performance anomaly detection*: identifying relatively rare cases where workload does *not* explain performance. Knowing when workload explains performance well vs. poorly can help to distinguish between true performance faults and mere overload. This in turn can inform our choice of performance analysis and debugging tools, and can also suggest remedial measures. Extensive empirical results demonstrate that the proposed method accurately models performance in three large distributed commercial production applications serving real customers. Furthermore it flags as anomalous an episode of a subtle performance bug in one of these applications.

SUMMARY

Users and providers of E-commerce sites and other Internet services value application-level performance. Unfortunately, performance in complex modern applications is difficult to understand. Analysis and remediation are easier if we can quickly determine whether performance is *surprising* given offered workload. If overload explains poor performance, adding resources might solve the problem; if not, re-starting components might be a reasonable expedient. If detailed diagnosis is required, knowing whether workload accounts for performance can guide our choice of tools: Overload might recommend bottleneck analysis, whereas poor performance that cannot be explained in terms of workload might suggest a fault in application logic or configuration.

This poster describes a simple model that relates application-level performance to offered workload. The model exploits four properties typical of commercially-important production applications: 1) workload consists of request-reply *transactions*; 2) transactions occur in a small number of *types* (e.g., "browse," "add to cart"); 3) resource demands vary widely *across* but not *within* transaction types; 4) resources are adequately provisioned, so *service* times (not *queueing* times) dominate response times. For applications with these properties, aggregate response time within a specified period is well explained as a weighted sum of per-type transaction counts, i.e., by models of the form

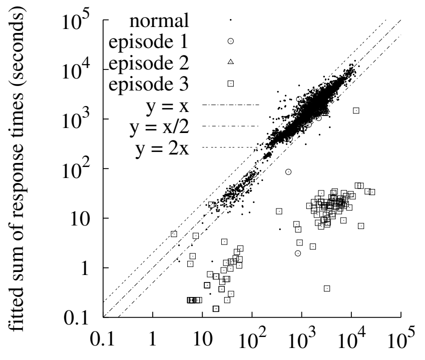
$$y_i \;\equiv\; \sum_j T_{ij} \;=\; a_1 N_{i1} + a_2 N_{i2} + \cdots \qquad (1)$$

where N_{ij} denotes the number of transactions of type j that began during interval i and T_{ij} denotes the sum of their response times. [More sophisticated models include additional terms for waiting times.] Input to the proposed method consists of N_{ij} and T_{ij}, suitably aggregated (this work uses 5-minute intervals); these may be obtained from an application transaction log, e.g., a Web server access log. Intuitively, parameter a_j represents the typical response time of individual type-j transactions.

Given parameters a_j and observed transaction counts N_{ij} during interval i, we substitute these into the right-hand side of Equation 1 to obtain a *fitted value* $\hat{y}_i$. The difference between $\hat{y}_i$ and observed value y_i is the *residual* $e_i = y_i - \hat{y}_i$. We measure model accuracy by the sum of absolute residuals $\sum_i |e_i|$ and employ a specialized linear programming technique to compute parameters a_j that minimize this quantity. (Ordinary least squares regression yields a substantially less accurate model; we report results for both parameter estimation procedures.)

Despite its simplicity, this method is remarkably accurate: fitted values $\hat{y}_i$ nearly always agree closely with observed y_i. An evaluation using three large and detailed data sets from distributed production systems serving real customers yielded normalized aggregate residuals $\sum_i |e_i| / \sum_i y_i$ between 14.7% and 19.7% for accuracy-maximizing a_j; least-squares regression yields aggregate errors that are between 3.2% and 9.5% higher. The distribution of relative error offers further insight into model accuracy: $|e_i|/y_i \leq 10\%$ in roughly 50% of cases and $\leq 20\%$ in roughly 75% of cases. As y ranges over four orders of magnitude, $\hat{y}$ almost never disagrees by more than a factor of two.

Large discrepancies between $\hat{y}_i$ and y_i are rare, but do such performance *anomalies* indicate performance *problems*? Empirical results suggest that sometimes they do. The scatterplot of $(y_i, \hat{y}_i)$ pairs below shows that an episode of a subtle performance bug in one application appears as prominent outliers in the lower right, whereas under normal conditions y_i and $\hat{y}_i$ agree closely. [Cohen *et al.* SOSP 2005 describes the bug.]



In conclusion, the performance of three large distributed applications serving real users is nearly always explained with remarkable accuracy by a simple and intuitive workload-based model. Furthermore, rare discrepancies between the model and observation sometimes correspond to true performance bugs. Ongoing work extends the method to improve accuracy, and tests the improved models in a wider range of applications.

SOSP'05 (poster), October 23–26, 2005, Brighton, United Kingdom.
ACM 1-59593-079-5/05/0010.