# A Provider-side View of Web Search Response Time

Yingying Chen Ratul Mahajan Baskar Sridharan Zhi-Li Zhang (Univ. of Minnesota)

Microsoft

**Abstract**— Using a large Web search service as a case study, we highlight the challenges that modern Web services face in understanding and diagnosing the response time experienced by users. We show that search response time (SRT) varies widely over time and also exhibits counter-intuitive behavior. It is actually higher during off-peak hours, when the query load is lower, than during peak hours. To resolve this paradox and explain SRT variations in general, we develop an analysis framework that separates systemic variations due to periodic changes in service usage and anomalous variations due to unanticipated events such as failures and denial-of-service attacks. We find that systemic SRT variations are primarily caused by systemic changes in aggregate network characteristics, nature of user queries, and browser types. For instance, one reason for higher SRTs during off-peak hours is that during those hours a greater fraction of queries come from slower, mainly-residential networks. We also develop a technique that, by factoring out the impact of such variations, robustly detects and diagnoses performance anomalies in SRT. Deployment experience shows that our technique detects three times more true (operator-verified) anomalies than existing techniques.

**Categories and subject descriptors:** C.4 [Performance of systems] Performance attributes; H.3.5 [Information storage and retrieval] Online information services

**Keywords:** Search response time; Web services; performance monitoring; anomaly detection and diagnosis

## 1. INTRODUCTION

Web services are the dominant enabler for a wide range of online activities such as searching and accessing content, shopping, and social interactions. Their performance is critical because even small increase in response time hurts user experience and impacts the monetization ability of service providers [22, 27]. It is thus extremely important for service providers to understand the key factors that impact performance and to quickly detect and diagnose any degradation. This task, however, is challenging because the performance of modern Web services depends on a variety of diverse, interacting factors that span servers in data centers, CDN edge servers, network paths, client machines and Web browsers, and user behavior.


*SIGCOMM'13,* August 12–16, 2013, Hong Kong, China.


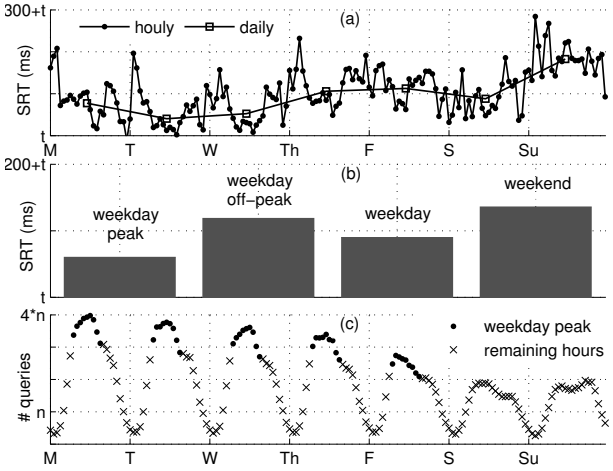


**Figure 1: Variation in SRT and query load at a large search provider.**

Using a large Web search service as an example, we highlight the challenges in understanding and diagnosing end-to-end search response time (SRT), that is, the delay between when the query is sent and when the response page is completely rendered. As an example, Figure 1(a) shows the average SRT for the service over the course of a week. Each data point is an average over all queries received in an hour from clients in the USA. We hide the absolute SRT values for confidentiality. We see that the SRT varies widely with hour of the day and day of the week. Surprisingly, as Figure 1(b) shows, it is higher during off-peak hours and weekends, when, as Figure 1(c) shows, the query load is in fact much lower. We have confirmed that this behavior is not due to operational practices such as switching off a subset of data center servers or changing routing policies during off-peak hours. We have also verified that this phenomenon is not unique to this service provider but holds for another large search provider as well.

To characterize and explain SRT variations, we develop an analysis framework that uses ideas from analysis of variance [15] and time series decomposition [26]. Our framework separates the observed SRT variation into two components: *i*) systemic variations that are caused by periodic changes in how the service is used, and *ii*) anomalous variations that are caused by unanticipated events such as server or network failures, network congestion, and denial-of-service attacks.

We use our framework to analyze over 6 months of data collected using detailed client- and server-side instrumentation. We find that the systemic variations in SRT can be attributed to three primary factors: the network characteristics of clients, the nature of queries, and the Web browser.

In contrast, the query processing time at the server has only a small impact on SRT variations.

The variations and interactions of the three primary factors explain almost all of the observed SRT variation, including the paradoxical behavior shown in Figure 1. Although the query load decreases significantly during off-peak hours, a greater fraction of the queries during these times come from slower, mainly-residential networks than faster, mainly-enterprise networks. Further, a greater fraction of the queries during off-peak hours result in media-rich response pages which take longer to download and render by browsers. This behavior likely stems from users performing more work-related queries (e.g., "correlation coefficient") during peak hours and more leisure-related queries during off-peak hours (e.g., "Lady Gaga" or "tourist attractions in Bali"). While the former class of queries tend to have text-based response pages, the latter class often has response pages with images and videos. Finally, a greater fraction of queries come from faster browsers during off-peak hours. However, the improvement in SRT due to this factor is not sufficient to counterbalance the impact of slower networks and richer response pages. The net effect is higher average SRT during off-peak hours and weekends.

The systemic variations in SRT make it difficult to detect and diagnose performance anomalies because the anomalies get masked by these variations. The operators of the search service, who employ visual inspection and conventional techniques (e.g.. outlier detection) to detect anomalies, inform us that it sometimes takes them multiple days to detect anomalies. Building on our analysis framework, we develop a technique for detecting anomalous variations in SRT, by factoring out the systemic variations, and for localizing where the root cause is likely to lie. We implement this technique as part of a tool that is deployed by the service. During the first five months of its deployment, it detects over 90% of the anomalies that operators detect using current practice. It also detects three times more true (operator-verified) anomalies than conventional techniques that do not account for systemic variations in SRT.

To our knowledge, our work is the first detailed analysis of variations in aggregate performance of a large-scale Web service. While it focuses on Web search, we believe our analysis framework and insights also apply to other Web services; their performance too is a function of variations in and interactions of user, network, and browser characteristics.

## 2. BACKGROUND: SEARCH SERVICES

Before describing our analysis, we provide a brief background on how modern Web services such as search are delivered. Figure 2 shows a typical infrastructure for such services. It consists of data centers that sit deep in the cloud and compute query responses, and of content distribution network (CDN) servers that sit close to the network edge and serve as intermediaries between users and data centers.

Each data center has a number of servers that are organized in tiers and play different roles in serving incoming queries. A tier-1 server, also known as a front door server, parses the request, invokes one or more tier-2 servers, and ranks and aggregates the answers from them into the response page that is sent to the user. Different tier-2 servers index different types of content such as Web pages, images, news, videos, and advertisements, and their answers are specific to that content. The front door server decides during aggregation which types of content answers are most relevant for a given query, based on hints provided by the tier-2 server, and includes only those answers in the response page. Due to their high cost, large data centers with all types of servers are located in only a few locations.

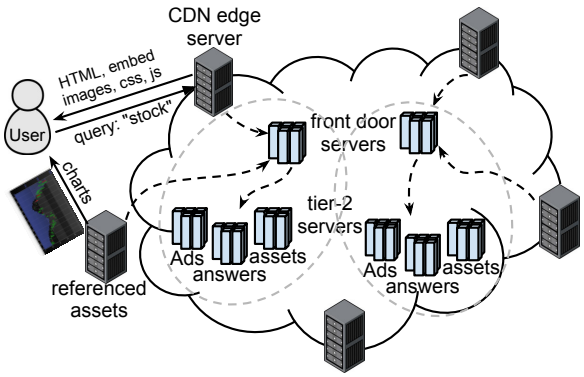


**Figure 2: Infrastructure of a large Web service.**

For performance, search services employ a large number of CDN edge servers. These servers help improve performance by terminating TCP connections closer to the user, which leads to a faster growth of the congestion window. They establish one or more long-lived, high-throughput TCP connections to front door servers and multiplex users on these connections. Due to the large diversity of queries across users and the personalization of responses, the CDN edge servers do not cache query results but fetch them from a data center. However, the results of popular queries may be cached in the data center.

User queries start with a DNS lookup, which returns IP addresses of one or more nearby CDN edge servers. The user then opens a TCP connection to an edge server and sends her query. Edge servers relay queries to a close data center and relay responses to the users.

Figure 3 shows the interaction between the front door server and the user, after abstracting out the CDN edge server. As soon as the query is received, the front door server relays it to various tier-2 servers. In parallel, it starts preparing the response which goes out in three chunks [29], shown as shaded areas in the figure. The first chunk contains the part of the result page that is the same across all types of queries, including the HTML header elements and the header portion containing the brand information (e.g., Google or Bing logo image). Personalized user information is also sent in this chunk.

The second chunk begins after the front door server has finished ranking and aggregating the results from the tier-2 servers. It contains the HTML portion of the response followed by BoP (bottom of page) javascript that is executed right after the entire HTML page is loaded, to refresh cookies, set image hovering properties, etc. The content of the chunk is typically compressed and must be decompressed by the browser before it can be parsed.

The response commonly contains pointers to additional *assets* that are needed to render the page. These include images, CSS, and more javascript. Some of these images are sent as objects *embedded* in the response itself. The front door server decides which images should be embedded and transmits them as the third chunk after fetching those images from their locations. The remaining assets, which

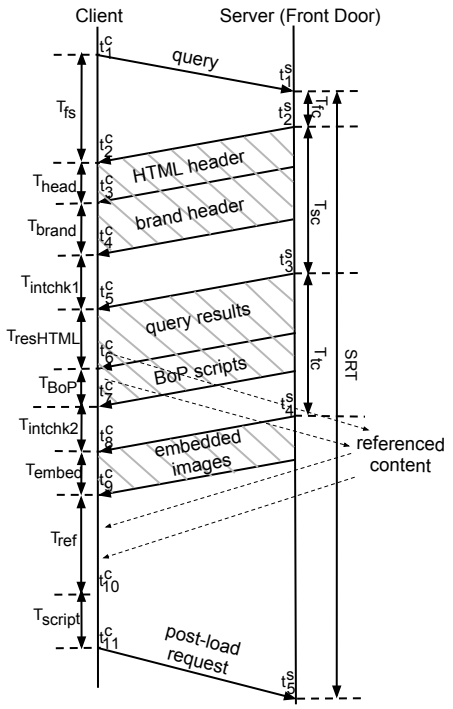


**Figure 3: Timeline of a search query.**

are *referenced*, are fetched by the user by issuing additional queries to servers that belong to the search provider or third-parties. These queries are pipelined and issued in parallel to the reception and parsing of the HTML content.

After all the content is received and objects in the page are fully loaded ($t^c_{10}$ in Figure 3), the browser fires the "onload" event and starts executing the javascript that corresponds to this event. After the script finishes executing ($t^c_{11}$), the browser finishes rendering the page and issues a post-load request for a tiny (1x1 pixel) image to the front door server using the same TCP connection. As described below, the time at which this request is received by the front door server is used to estimate the SRT for the query.

# 3. DATA COLLECTION

To gain insight into SRT behavior and factors that impact it, we collect detailed data from one of the largest search providers in the world. The granularity of the data is at the level of individual queries. For each query, a set of server-side and client-side metrics are collected. The server-side metrics include all timestamps shown on the right in Figure 3 ($t^s_*$). The client-side metrics include the type of browser (user-agent) and the timestamps on the left ($t^c_*$), which are collected using javascript and HTML instrumentation. The timestamps are taken by the browser and returned to the server as part of the post-load request (at $t^c_{11}$). Through controlled experiments, we have verified that the overhead of collecting this data, including its impact on SRT, is negligible.

We use the collected timestamps to compute the time delay measures ($T_*$) that are shown in the figure. Each measure captures the time taken for a certain activity to complete on the client or the server side. For instance, $T_{fc}$ is the time it takes for the front door server to transmit the first byte of the first chunk after receiving the query and $T_{intchk1}$ is the time the client has to wait between the last byte of the first chunk and the first byte of the second chunk.

We approximate three delay measures that cannot be directly computed from the timestamps. The first is SRT itself. While considering SRT, we ignore the time for performing the DNS lookup and establishing the TCP connection with the CDN edge server. Both these activities are relatively quick as they involve one round trip to a nearby server and DNS names are often cached. (We have empirically verified this fact using a separate data source.) Ignoring these two factors, the SRT for a query is $t^c_{11} - t^c_1$. But our data does not have $t^c_1$ because HTML instrumentation does not allow us to log the time when the query is sent. We thus estimate SRT as $t^s_5 - t^s_1$, which closely approximates $t^c_{11} - t^c_1$ when the one-way delay of the original query is similar to that of the post-load query [2, 24].

The second measure that we approximate is network round trip delay, $T_{net}$ (not shown in Figure 3). We approximate it as $T_{net} = (t^s_5 - t^s_2) - (t^c_{11} - t^c_2)$. The first parenthetical quantity is the time between the server starting to send the first chunk to receiving the post-load query, and the second parenthetical quantity is the time between the client starting to receive the first chunk and sending the post-load query. Their difference approximates the round trip network delay. It includes the time taken by CDN edge servers to relay queries and responses between clients and front door servers. Data from the CDN edge servers confirm that this time is negligible compared to network delays; relaying involves minimal processing and uses passthrough techniques.

The third measure that we approximate is $T_{fs}$, the time it takes for the first byte of the first chunk to arrive at the client. We approximate it as $T_{fs} = T_{fc} + T_{net}$.

In addition to the query-level metrics, we obtain from our CDN information about the clients. This includes the client's location and the ASN (Autonomous System Number) of its ISP. The CDN also provides us a coarse estimate of the client access link bandwidth, categorized into six buckets: [1, 56], [57, 256], [257, 999], [1000, 1999], [2000, 5000], and [5000, $\infty$] Kbps. As the access bandwidth of a client is not susceptible to fast change, our CDN measures it once every two months.

The data used in this paper was continuously collected for a span of over 6 months. We only consider Web search, which is the most common type of search. To focus on clients that experience roughly similar conditions, we exclude queries from outside the USA and from mobile devices. These clients experience disparate conditions and the responses to their queries are also different. In compliance with confidentiality constraints, all results in this paper are anonymized and normalized.

# 4. SYSTEMIC SRT VARIATIONS

To identify the key factors that contribute to SRT variations and dissect their complex interactions, we develop an analysis framework that helps separate the observed SRT variations into systemic and anomalous variations. In this section we present our framework, with a focus on systemic variations. Detecting and diagnosing the anomalous variations will be discussed in the next section.

## 4.1 Analysis Framework

The observed SRT variations are likely due to a confluence of complex, interacting factors such as network bandwidth,

| Measure | Impact factors |
|---|---|
| $T_{fs}$ | network, server |
| $T_{head}$ | browser, network |
| $T_{brand}$ | browser, network |
| $T_{intchk1}$ | query, server |
| $T_{resHTML}$ | browser, query, network |
| $T_{BoP}$ | browser, query |
| $T_{intchk2}$ | query, server |
| $T_{embed}$ | query, network |
| $T_{ref}$ | browser, query, network |
| $T_{script}$ | browser, query |
| $T_{fc}$ | server |
| $T_{sc}$ | query, server |
| $T_{tc}$ | query, server |
| $T_{net}$ | network |

**Table 1: Factors that impact each measure.**

end-to-end latency, server-side processing time, and browser speed. To isolate and identify the key contributing factors, we start by decomposing the overall SRT into individual measures shown in the first column of Table 1. See Figure 3 for what each measure represents. The relationship between individual measures and underlying factors of interest (e.g., network) is in general not one-to-one. As shown in Table 1, most measures are impacted by multiple factors, and each factor impacts multiple measures. A factor may also influence different measures in different degrees (i.e., the effects are unbalanced). These complex dependencies make it hard to tease out the factors that cause SRT variations; we cannot simply correlate SRT to measures that cleanly capture individual factors. We describe below how we identify which factors cause most variation and quantify their impact.

### 4.1.1 Methodology

Let $Y$ be a random, response variable that represents the observed SRT, and let $X_k$, $k = 1, 2, \ldots, n$, be a set of random, explanatory variables. Each $X_k$ represents one of the constituent measures in Table 1. We assume a linear model $M := Y = a_0 + \sum_k a_k X_k + \eta$, where $\eta$ represents random noise in measurements. This model is an approximation because the dependence between $Y$ and $X_k$ may not be linear in practice. Further, because of dependencies between explanatory variables, the model may not have a unique solution. But we find this simple model to be useful in analysis and we are careful to account for variable interaction when using the model. We use measurement data to learn $a_0$ and $a_k$'s that best match the model $M$.

Given this model, our analysis proceeds in three steps. First, we separate the variance due to random noise in the measurement from the the variance captured by the model. The latter is what we refer to as systemic variance. Second, given the systemic variance thus extracted, we quantify the contribution of individual measures $X_k$, which in turn lets us identify the primary factors that cause SRT variation. Finally, we investigate the contribution of each primary factor by controlling the other primary factors. In this subsection (§4.1), we described the first two steps, followed by their results. Investigations that correspond to the final step are in §4.2–4.4.

In the first two steps of our analysis, we apply ideas from analysis of variance (ANOVA) [15]. The first step starts by computing the variance in the response variable, $SS_T(Y) = \sum_i (y_i - \bar{y})^2$, where $y_i$ is an individual (independent) observation of the response variable and $\bar{y}$ is the mean value across all observations. We then partition $SS_T(Y)$ into $SS_R(M, Y)$ and $SS_E(M, Y)$, i.e., $SS_T(Y) = SS_R(M, Y) + SS_E(M, Y)$, where $SS_R(M, Y)$ is the variance in $Y$ explained by the model $M$ and $SS_E(M, Y)$ is the variance that is not explained by the model.[1] Let $\hat{y}_i = a_0 + \sum_k a_k x_{ki}$, where $x_{ki}$ is an individual observation of the explanatory variable $X_k$. Then $SS_R(M, Y) = \sum_i (\hat{y}_i - \bar{y})^2$, and $SS_E(M, Y) = \sum_i (y_i - \hat{y}_i)^2$.

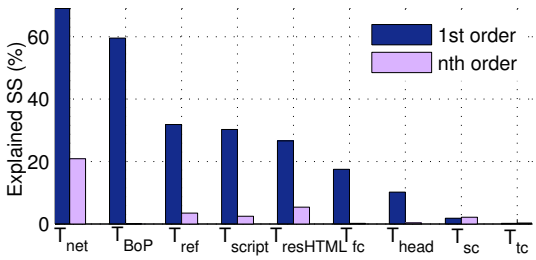


**Figure 4: Analysis of variance results.**

In the second step of our analysis, we partition the model variance $SS_R(M, Y)$ into $n$ components, each of which is attributable to an explanatory variable and helps quantify the extent to which that variable explains SRT variance. For this purpose, we focus on two important metrics:

*i) 1st order variance* $SS_R(M_k, Y)$: It quantifies the variance explained by only one explanatory variable $X_k$, as if the SRT depends on only one variable. That is, $SS_R(M_k, Y)$ is the model variance of the model $M_k := Y = b_0^k + b^k X_k + \eta$. Formally, $SS_R(M_k, Y) = \sum_i (b_0^k + b^k x_{ki} - \bar{y})^2$, where values for $b_0^k$ and $b^k$ are learned from the measurement data.

*ii) n-th order variance* $SS_R(M_{\neg k}, Y)$: It quantifies the left-over variance that is explained by $X_k$ but cannot be explained by the interactions of the other $n - 1$ variables. That is, $SS_R(M_{\neg k}, Y)$ is the model variance of the model $M_{\neg k} := Y = c_0^k + \sum_{j \neq k} c_j^k X_j + \eta$. Formally, $SS_R(M_{\neg k}, Y) = \sum_i (c_0^k + \sum_{j \neq k} c_j^k x_{ji} - \bar{y})^2$, where values for $c_0^k$ and $c_j^k$'s are learned from the measurement data.

We apply the steps above on 1-hour averages of SRT and the individual delay measures listed in Table 1. For computational efficiency and minimizing redundancy, we consider only those individual measures that have a noticeable impact on SRT. Specifically, we exclude from our analysis measures that either constitute a minuscule fraction of SRT or are highly correlated with another measure. The first criterion excludes $T_{intchk2}$ and $T_{embed}$ because they represent less than 1% of the SRT. The second excludes $T_{fs}$, $T_{intchk1}$, and $T_{brand}$ because they are highly correlated with, respectively, $T_{net}$, $T_{sc}$, and $T_{resHTML}$; the Pearson's correlation coefficients are 0.99, 0.79, 0.99. (High correlation between $T_{fs}$ and $T_{net}$ indicates that $T_{fs}$ is largely determined by network latency, and the server-side processing time to generate the first chunk ($T_{fc}$) has only a minute effect.)

### 4.1.2 Results: Primary factors

Figure 4 shows the amount of variation in SRT that is explained by each of the 9 remaining measures. Focusing on the 1st order variance first, we see that the two measures that explain the most variations, roughly 60% each, are $T_{net}$ and $T_{BoP}$. $T_{net}$ is impacted by the network latency between the client and the (tier-1) server. $T_{BoP}$ is impacted

[1] In this notation, $SS$ is short for sum of squares. The subscripts $T$, $R$, and $E$ are short for total, regression, and error.

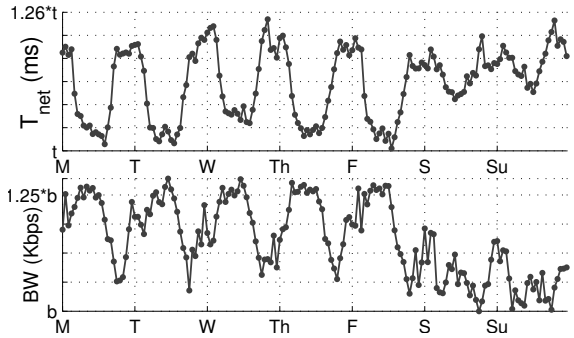

**Figure 5: Variation in network characteristics.**

mainly by the browser speed and query type (e.g., results pages with more images have more complicated bottom-of-page scripts). The next three measures explain roughly equal amounts of variations, around 30% each. Of these, $T_{resHTML}$ and $T_{ref}$ are impacted by network latency and bandwidth since they involve downloading, respectively, of the results page and referenced content, by the query type since size of the results page and referenced content depend on query, and by the browser's speed of rendering the results and referenced content. $T_{script}$ depends mainly on the browser's speed of javascript execution.

The measures that are impacted by the server processing time, $T_{fc}$, $T_{sc}$, and $T_{tc}$, explain relatively small amount of variations in SRT. (Some of these measures are impacted by the query type as well.) Thus, all measures that explain significant SRT variations are impacted by network, query, and browser, but not by the server-side processing time.

Interpreting these results requires some care. Our results do not imply that server-side processing time is not an important contributor to the total SRT. Server time could be a big fraction of SRT and yet not be responsible for significant systemic variation because of its relative stability. This stability could stem from techniques in the data center that generate response within a configured deadline (e.g., 250 ms) [6, 10]. Further, since we study average SRT across all queries in an hour, our results also do not imply that server processing time is stable at the level of individual queries. Various short-lived issues (e.g., OS-level scheduling) could lead to high server delays for individual queries [13] and yet those delays may not be a factor behind systemic variations across hours of the day.

Next, looking at the n-th order variance, we see that most variables explain only a small amount of variance that cannot be explained by other variables. This underscores the high degree of interaction among various variables. $T_{net}$ is the only variable with notable n-th order variance.

Though not shown in the figure, we also find that collectively these measures capture almost all of the variance in SRT. The amount of SRT variations that cannot be explained by them (i.e., $SS_E$) is only 0.66%.

In summary, we find that the systemic variations in SRT stem primarily from network characteristics, query type, and browser speed; server-side processing time has a relatively small impact. That these three factors lead to systemic SRT variations implies that they must be systemically varying across times of day and days of week. The following sections investigate how and why these three factors vary.

## 4.2 Variation in Network Characteristics

We begin by studying variation in network characteristics. We show that it stems from changes in the relative fraction of queries that come from residential and enterprise networks.

Figure 5 shows the network characteristics observed across all of our clients. For a week-long period, it plots the hourly average of network latency ($T_{net}$) and the bandwidth (BW) reported by our CDN. All the graphs in this paper that show a week's worth of data correspond to the same week; unless there is a weekday holiday, other weeks have similar behaviors. We see the network characteristics do vary systemically over time. During off-peak hours, network latency increases by as much as 20% and the bandwidth decreases by a similar percentage. These systemic changes explain why network characteristics is a key factor underlying SRT variations.

Now, the question is why network characteristics vary as shown. If anything, we would have expected the opposite: due to possible congestion during peak hours, network latency should have been higher during peak hours. As we explain below, the variations we see are due to the variations in the fraction of queries that come from residential networks which tend to have higher latencies and lower bandwidths compared to enterprise networks.

We use a simple heuristic to classify a network as residential or enterprise, based on the expectation that enterprise networks send relatively few queries during weekends. For each of the 13,349 ASNs observed in our data, we compute the ratio of the number of queries they send during the weekend days (i.e., Saturday and Sunday) to the number during weekdays. Across all ASNs, this ratio varies between 0 and 10. For a conservative classification, we deem one-third of the ASNs with the lowest ratios as *mainly-enterprise* and one-third of the ASNs with the highest ratios as *mainly-residential*. The middle third is deemed as *mixed-or-unknown*. The weekend-to-weekday query ratio of all mainly-enterprise ASNs is lower than 0.0074. We do not expect the ratio to be perfectly zero for many enterprises if some employees work on the weekends or connect their laptops through corporate VPNs (Virtual Private Networks). We manually verified the classification of many ASNs that are deemed mainly-residential or mainly-enterprise, based on their names. For instance, Microsoft (ASN 3598) is classified as mainly-enterprise.[2]

With this classification in place, we can now explain the variation in network characteristics. As shown in Figure 6, the mainly-residential ASNs send a greater proportion of queries during off-peak hours (left graph) and they have poorer network characteristics (middle two graphs). Their share is 70% during peak hours but almost 100% during the off-peak hours, and their network latency is 25% higher than that of mainly-enterprise ASNs.

The poorer network characteristics of mainly-residential ASNs translate, expectedly, to higher SRTs (right graph). On average, the SRT for these ASNs is 11.2% higher than that of mainly-enterprise ASNs. Due to the low traffic volume from mainly-enterprise ASNs during off-peak hours, we only plot their performance during weekday peak hours. To illustrate this behavior without conflating with the impact of browser speed and query type on SRT, the graph is plot-

[2]We tried initially to use ASN names to classify ASNs but dropped that effort because of the names of many ASNs are hard to interpret.

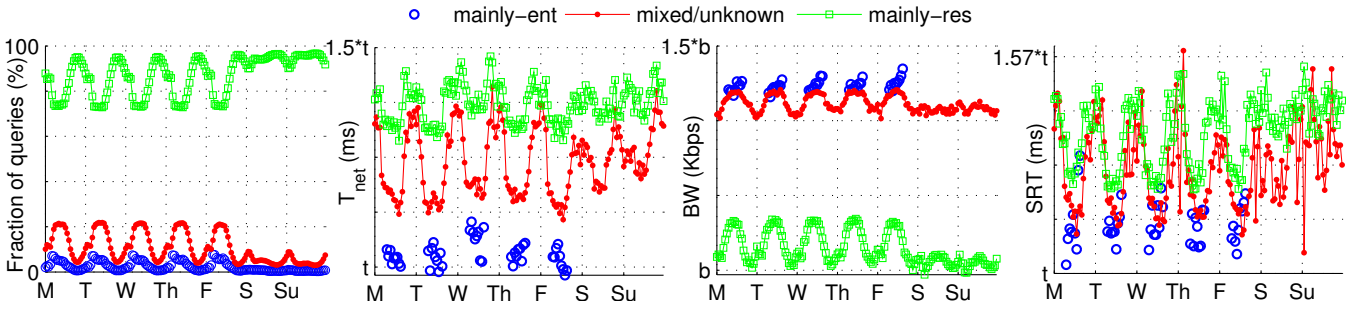


**Figure 6: Comparing mainly-enterprise, mixed-or-unknown, and mainly-residential ASNs.**

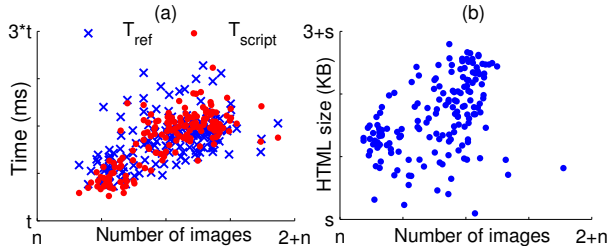


**Figure 7: Image count vs. other measures of response page complexity.**

ted using data that corresponds to only one type of browser and queries that generate similar response pages (specifically, pages without images). Other browsers and query types show a similar effect.

## 4.3 Variation in Query Type

We now investigate the variation in query type. The nature of the query can impact SRT in two ways. First, it can impact the time it takes the server to compute the results. Second, because different queries have different response pages, it can impact the time it takes to download all content (which includes HTML and multimedia content) and for the browser to load the page (which includes javascript execution and rendering). We showed earlier that server processing time is not a primary factor in SRT variations, and detailed measurements based on different types of queries confirms that the variation in server processing time is small.

Thus, variations in SRT due to query type must largely stem from the diversity and variations in type of responses generated. While most queries generate ten Web answers (HTML links), depending on the query, the response page can also contain other types of answers such as news, images, videos, and maps. These differences lead to high degree of diversity in response pages. The number of HTML bytes in the response page varies by an order of magnitude across individual queries, and the number of embedded or referenced images vary by more than a factor of two. Consequently, different response pages take different times to download, parse, and render.

Though researchers have recently proposed general measures of Web page complexity [11], we use a simple metric that is specific to the domain of search. Because most non-Web answers contain images, we use the number of images on the response page as a metric for page complexity. It turns out that, as shown in Figure 7, this metric is correlated with other possible measures of page complexity such as time to download all referenced content ($T_{ref}$ in left graph), time to load the page after all content has been downloaded ($T_{script}$ in left graph), and HTML size (right graph).

Figure 8 shows that the number of images in query responses varies systematically across hours of day and days of week. Further, Figure 9 shows that the SRT is higher for queries with more images in their responses. The spread in SRT values is about 30%. To factor out the impact of network and browser type, this figure is plotted for one type of browser and mainly-residential ASNs. We obtain qualitatively similar results for other browser and ASN types.

Taken together, the two effects—variation in query richness and its impact on SRT—explain why the nature of the queries is a primary factor behind SRT variations.

One interesting question is *why* query richness varies with time. An in-depth look at the data reveals that richer queries tend to be leisure-oriented such as queries about celebrities, holiday events, and travel destinations. Users tend to issue more of such queries during off-peak hours or when they are at home. As shown in Figure 10, the number of images contained in the queries from mainly-residential ASNs are higher than those from mainly-enterprise ASNs. Therefore, the shift of the "user intent" from peak to off-peak hours is what leads to variations in query richness.

## 4.4 Variation in Browser Types

We now show that browsers are a primary factor behind SRT variations because *i*) the relative mix of browsers accessing the search service varies over time; and *ii*) different browsers have different speeds.

There are eight types of browsers in our data that issue at least 1% of the queries. Different major versions of the same browser (e.g., Internet Explorer 8 and 9) are considered different types of browsers because they can have significantly different rendering and script execution engines.

Figure 11 shows the variations of the fraction of queries for the two most popular browsers, Browser-X and Browser-Y. These two browsers account for 35% and 40% of the total queries, respectively. We see that the fraction of queries from these browsers varies substantially with time. Their relative popularity swings by over 25%: Browser-X goes from generating 15% more queries during peak hours to generating 10% fewer queries. The relative popularity of other six major browser types varies as well. Two of them vary like Browser-X, i.e., they send a higher fraction of their queries during peak hours. The remaining four vary like Browser-Y.

In addition to generating different fractions of queries over time, different browsers have disparate performance. As an indicator of browser performance, we use $T_{script}$, which is not impacted by the time it takes to download the page be-

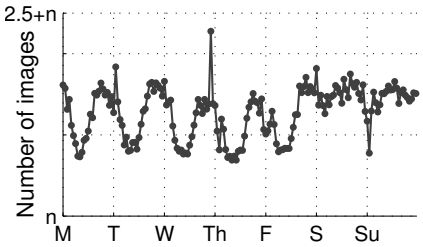


**Figure 8: Variation in the image count in response pages.**

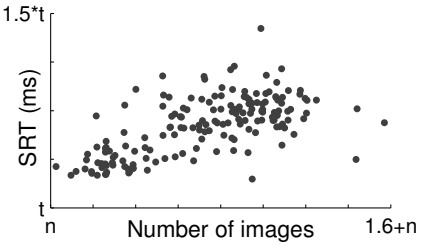


**Figure 9: SRT vs. image count in response pages.**

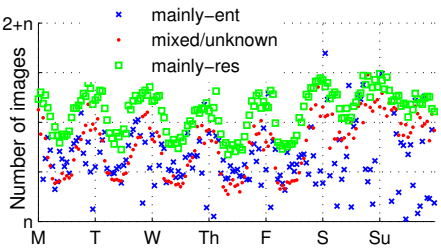


**Figure 10: Image count variation in different networks.**

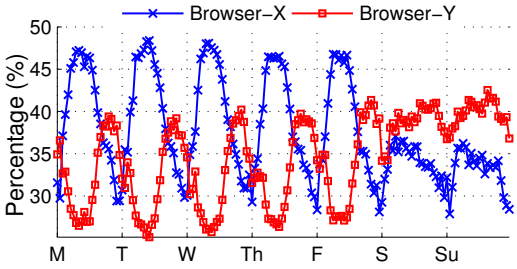


**Figure 11: %-age of queries from the top two browsers.**

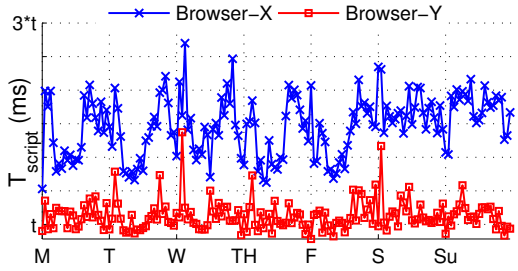


**Figure 12: $T_{script}$ for the two major browsers.**

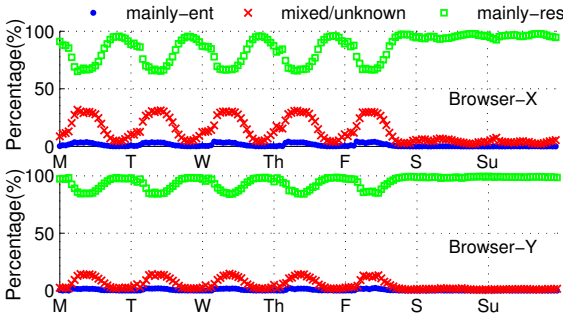


**Figure 13: %-age of queries from diff. nets.**

cause it captures the time from downloading all content to fully loading the page. Figure 12 shows the performance of the two popular browsers. On average, Browser-X is slower by a factor of 1.82. To minimize the impact from network and query type, we only plot it for queries whose response pages do not contain images and come from mainly-residential ASNs; other types of queries and ASNs show a similar behavior. As we can see from the figure, the two browsers have disparate performance. Controlled experiments on the same machine with different browsers confirm that the differences in $T_{script}$ times that we see in the wild reflect real differences in browser performance, and not just differences in the capabilities of host machines.

The combination of the two observations—variations in the browser mix and the differences in browser speed—explains why browsers are a primary factor behind SRT variations.

To understand *why* the mix of browsers varies with time, we investigated where their users come from. Figure 13 shows the fractions of queries coming from three different categories of networks for the two popular browsers. We see that the vast majority of the Browser-Y's queries come from mainly-residential networks. For Browser-X, however, there is a significant shift (around 40%) during peak hours away from mainly-residential networks. We speculate that these trends stem from the fact that Browser-X, at the time of this study, is more likely to be the standard browser that is adopted and supported officially by many enterprises. This speculation is supported by the fact that Browser-X is an older browser, while Browser-Y is newer; many enterprises in the USA do not immediately upgrade to newer browsers as they must first test compatibility of the new browser with their internal services (called line-of-business applications). On the other hand, residential machines in the USA are likely upgraded sooner due to automatic updates. Thus, as the users of the search service move from work to home, the service sees a move from Browser-X to Browser-Y.

Observe that the impact of browsers on SRT is the opposite of the impact of network and query type. During off-peak hours, a greater fraction of queries come from Browser-Y, which has better performance. This should have a positive impact on SRT during off-peak hours. But the negative impact on SRT due to variations in network and query type dominates, and we see higher SRTs as the net effect. Without the corrective effect of browsers, the SRTs during off-peak hours would have been even higher.

## 5. ANOMALOUS SRT VARIATIONS

In addition to systemic variations that we study above, SRT also experiences irregular variations. These variations, which we call anomalies, stem from events such as failures in the network or data center, congestion, and attacks on the search infrastructure. For a good user experience, anomalies must be detected and resolved quickly. However, operators inform us that it frequently takes them several days before they even detect that the SRT is anomalous, as the systemic variations often hide the real anomalies. Once detected, diagnosing the root cause of an anomaly is also challenging as it can lie in any part of the infrastructure.

We develop a technique to assist the operators in quickly detecting and diagnosing SRT anomalies. Below, we first describe how it detects anomalies and then how it localizes their root causes. We have implemented our technique in a tool that has been deployed on the real system.

## 5.1 Detecting Anomalies

The main challenge in accurately detecting SRT anomalies is that they co-exist with systemic variations due to other factors, including the weekly, daily and hourly fluctuations, and the long-term evolution of SRT. As we show later, due to these variations, common anomaly and outlier detection methods both fail to detect many anomalies and flag events that are not anomalies.

### 5.1.1 Methodology

We use an approach based on time series decomposition [26], which we call WoW (week-over-week) analysis. The basic idea is to view SRT as a composition of three components: *i*) the long-term trend; *ii*) the seasonality or periodic behavior; and *iii*) fast variations or noise.

Consider the SRT time series, where $SRT_t$ is the average over the $t$-th hour. The long-term trend component, denoted by $L$, of this series can be computed as a centered moving average with the window size set to $T$:

$$L_t = \frac{1}{T+1} \sum_{i=-T/2}^{T/2} SRT_{t+i} \quad (1)$$

$T$ should be greater than or equal to the maximum periodicity in the data. We use $T$=168 hours (1 week).

Let $Y_t$=$SRT_t$–$L_t$ be the time series after removing the long-term trend. Then, the seasonal component, $S$, can be computed as seasonal moving average:

$$S_t = \frac{1}{M+1} \sum_{i=0}^{M} Y_{t-iT}, M = \lfloor t/T \rfloor \quad (2)$$

What remains now is the noise component, N, which can be computed by removing the long-term trend and seasonality components:

$$N_t = SRT_t - L_t - S_t \quad (3)$$

By definition, this component is neither part of the long term trend nor a periodic event. It captures the irregularity that cannot be explained by the other two factors.

We deem as anomalous time instances where the noise is abnormally high. To infer high abnormality, we assume that noise follows a Gaussian distribution based on the central limit theorem [1]. However, due to the diurnal patterns and the day of week effect, the distribution parameters can differ for different times of week.

Thus, based on historical data, we learn 168 Gaussian models, one for each hour of the week. We flag as anomalous values outside the 95th percentile of the distribution (i.e., $\pm$1.96). This threshold is commonly used in many statistical areas [25] and we find that it works well in our setting too. Thus, we deem that the SRT at time $t$ is anomalous if:

$$\frac{|N_t - \mu_{\hat{t}}|}{\sigma_{\hat{t}}} > 1.96 \quad (4)$$

where $\mu_{\hat{t}}$ and $\sigma_{\hat{t}}$ are the mean and standard deviation of the Gaussian distribution built from all data collected at the same hours as $t$ within a week.

The quantity on the left in Eq. 4 is the *severity* of the anomaly, and we report it in the notifications that are generated for the operators. It captures the extent to which current SRT has deviated from expectation. Larger values indicate more serious anomalies.

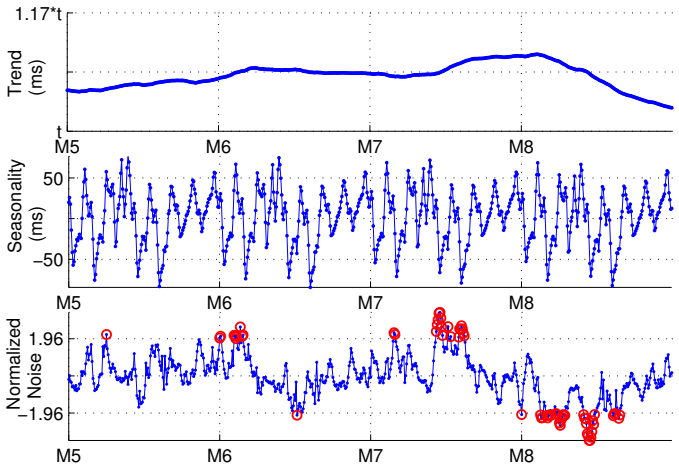


**Figure 15: Time series decomposition results. Circles in the bottom graph denote anomalies.**

We repeat the above steps for each incoming data point to conduct online anomaly detection. If an hour is anomalous, its data is excluded while learning the Gaussian model. We find that including data from anomalous hours leads to less robust anomaly detection.

### 5.1.2 Results

To evaluate our method, we compare it to two common methods for anomaly detection that do not account for the interference from systemic variations. The first method builds a single Gaussian model of SRT using recent history, and it detects anomalies when the current value is an outlier. The second method is based on change point detection [30]. To eliminate the potential trend change, sudden changes in SRT are detected as change points using cumulative sum and bootstrapping techniques [30]. A Gaussian model is built for all the hourly data points between each pair of change points. In both methods, a value $v$ is deemed anomalous if $\frac{|v-\mu|}{\sigma} > 1.96$, where $\mu$ and $\sigma$ are the parameters of its learned model.

We first illustrate the behavior of the three approaches and then quantify their overall performance. For an examplary four-week period, Figure 14 shows the anomaly detection results (overlaid on top of SRT) for each approach. Figure 15 shows the time series decomposition results of WoW analysis for the corresponding period.

As we see in Figure 14, the three approaches do behave differently. The simple Gaussian model approach only detects globally huge spikes, which can overlap with the systemic variations. The change point approach detects only local spikes between any two change points. The WoW approach can detect not only globally huge spikes, but also anomalies that are not visually apparent due to interference from systemic variations. As one example, for 5AM on the 3rd Friday, an anomaly was detected by WoW approach but was missed by the other approaches. This was a real anomaly; historical behavior of SRT, shown in Figure 16, reveals that the SRT at 5AM on Fridays is generally much lower.

The conventional approaches are fooled by systemic variations from the other perspective too. That is, they flag non-anomalies. As one example, these approaches detected an anomaly at 6PM on the 3rd Monday in Figure 14, but WoW did not flag that time period. As the historical data

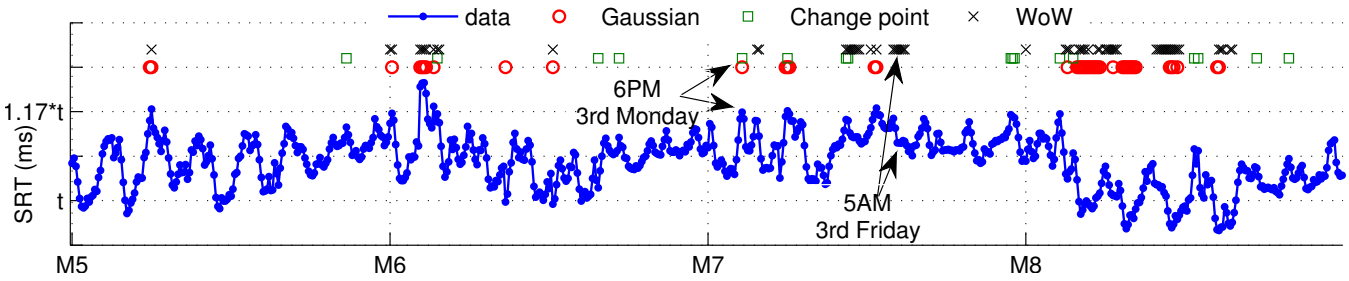


**Figure 14: Comparing anomaly detection results for three different techniques.**

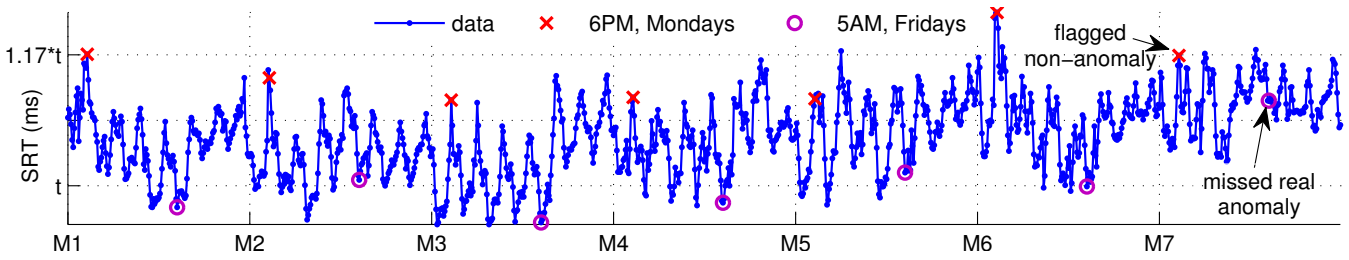


**Figure 16: Historical behavior for the anomalies flagged or missed by Gaussian and change point techniques.**

in Figure 16 displays, this time period does not have significantly degraded SRT compared to 6PM on past Mondays.

We now perform a more systematic evaluation. The performance of anomaly detection can be quantified using false negative and positive rates. False negatives are cases in which a real anomaly is missed, and false positives are cases in which a non-anomaly is flagged. The results below are based on five months of data.

**False negatives:** Quantifying the false negative rate requires as a reference a complete list of all anomalies in the system. However, such a list is rarely available for a complex, real system. We use instead the ticket database that is maintained by the search provider. The anomalies documented in this database have been manually detected by the operators based on visual inspection of the data or user complaints, or they have been flagged by an existing tool (which does not account for SRT variations) and later verified manually.

We find that 90%, 65%, and 60% of all the anomalies present in this database are identified by WoW, Gaussian, and change point approaches. That is, while WoW missed 10% of the anomalies, the other approaches miss 3-4 times as many.

Comparing the ticket database and our tool, we find that the anomalies in the database tend to have high severity values ($> 2.5$, or outside the 99th percentile of the Gaussian model). This implies that with the current practice, operators could detect only highly anomalous events. Further, our tool flags many anomalies that are not in the ticket database. If these anomalies are not false positives, which we study next, they represent anomalies that the current practice misses.

**False positives:** Estimating the false positive rate of an anomaly detection tool is challenging. Investigating an anomaly can take a huge amount of effort and thus operators do not investigate each anomaly. For instance, short-lived anomalies that disappear before an operator has had a chance to investigate are not investigated. (However, operators still want to detect and log all anomalies; this record helps quantify service reliability and determine if some components fail repeatedly.) Thus, we cannot be sure if a certain anomaly that was flagged by our tool but not investigated is a true or a false positive.

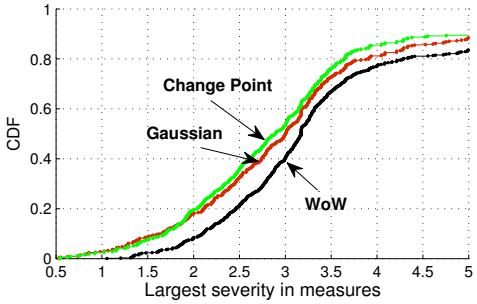


**Figure 17: Distribution of largest severity across all measures.**

To estimate the false positive rate, we emulate the method used by the operators as a first step towards investigating an anomaly. The operators observe the behavior of other, fine-grained measures (e.g., $T_{net}$) during the anomaly period. If one or more of those measures is anomalous as well, the anomaly is deemed as likely real and further investigation is conducted. If none of those measures is anomalous, the anomaly is deemed as false. The assumption is that if the anomalous behavior in SRT correlates to anomalous behavior in at least one of the fine-grained measures, then we can be confident that this is a real anomaly.

Thus, we apply the same WoW technique to the 14 fine-grained measures (§4) and compute the largest severity value across all measures. Figure 17 shows the distribution of the largest severity value as an SRT anomaly is detected using three different techniques. We consider an SRT anomaly to be a false positive, if the largest severity value does not indicate an anomaly, that is, it is under 1.96. Based on this criteria, we see that the false positive rate of WoW is

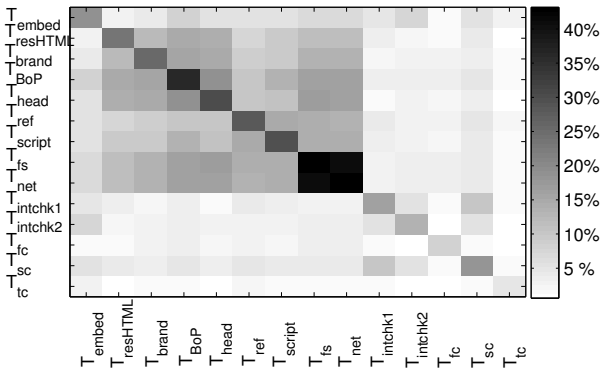


**Figure 18: %-age of anomalies for pairs of measures.**

7%, while that of Gaussian (17%) and change point (19%) approaches is at least twice as much.

## 5.2 Diagnosing Anomalies

In addition to detecting anomalies, our tool helps operators diagnose those anomalies, by identifying at a coarse-level the most likely source of the problem. For our purposes, possible sources are client behavior, the data center, and the network between the clients and the data center (which includes the CDN servers). Client behaviors can be a source of anomalies due to attacks (e.g., bots generating a lot of queries), among other possibilities.

The inference of our tool is then combined with other lower-level measures (e.g., output of tools that monitor network or server health and utilization) to localize the root cause at a finer granularity. These low-level measures are by themselves insufficient for root cause localization as they are noisy and their impact on SRT is otherwise unclear [9].

### 5.2.1 Methodology

The anomaly diagnosis functionality of our tool is invoked whenever an SRT anomaly is detected. Its starting points are the time series of the 14 measures that we used to study systemic SRT variations.

We face two main challenges. First, due to complex interactions among different measures, simply using the anomaly severity value of individual time series does not suffice. For example, when an anomaly happens in the network, both $T_{resHTML}$ and $T_{net}$ can be anomalous. Second, during anomalies, the relationships among the different measures may vary from the normal case. For example, under normal circumstances, $T_{fs}$ (which is $T_{net} + T_{fc}$) and $T_{net}$ are highly correlated usually, but if $T_{fc}$ is anomalous due to a failure in the data center, this correlation disappears.

Thus, a better understanding of the relationships among the various measures *during* SRT anomalies is important. To obtain insight into these relationships, we investigate how often two measures are simultaneously anomalous during SRT anomalies. Figure 18 shows the percentage of SRT anomalies in which a pair of measures, specified on the x- and y-axis, appear as anomalies at the same time. The diagonal values are the percentage of SRT anomalies when the measures, specified on the x (or y)-axis, appear as an anomaly by themselves. The graph is symmetric along the diagonal.

We make the following observations from this figure. First, focusing on $T_{net}$, a pure network-side measure, we see that $T_{fs}$ is almost always anomalous along with it. The server-side measures ($T_{fc}$, $T_{sc}$, $T_{tc}$) and client-side measures that

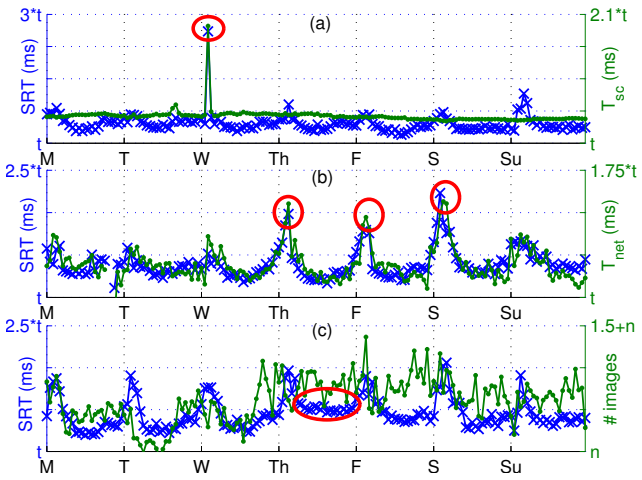


**Figure 19: Example anomalies.**

are highly dependent on the server behavior ($T_{intchk1}$ and $T_{intchk2}$) are rarely anomalous in conjunction with $T_{net}$. Other browser-side measures are sometimes anomalous in conjunction with $T_{net}$. Second, let us focus on pure server-side measures ($T_{fc}$, $T_{sc}$, $T_{tc}$). They appear anomalous mostly on their own, except for *i*) $T_{intchk1}$, which is a browser side measure that is highly dependent on server-side latency; and *ii*) $T_{intchk2}$ and $T_{embed}$, which we can ignore as they contribute less than 1% of the total SRT. Third, the anomalies in browser-side measures ($T_{resHTML}$, $T_{brand}$, $T_{BoP}$, $T_{head}$, $T_{ref}$, and $T_{script}$) are weakly correlated with server- and network-side measures, indicating their anomalies likely stem from client issues.

Based on the observations above, to diagnose anomalies, we first divide the measures into three classes: Network ($T_{net}$ and $T_{fs}$), Server ($T_{fc}$, $T_{sc}$, $T_{tc}$), or Client ($T_{resHTML}$, $T_{brand}$, $T_{BoP}$, $T_{head}$, $T_{ref}$, and $T_{script}$). If only one class of measures is anomalous, then we deem the problem likely lies in the corresponding class. We saw above that, frequently, only one class is anomalous when SRT is anomalous.

If measures in more than one class are anomalous, we use the following decision logic to determine the likely root cause (or prioritize the investigation). First, if any Server measures are anomalous, we deem that the anomaly is in the data center. This heuristic is based on the second observation above and the fact that the server-side measures are collected using a separate instrumentation (not Javascript), which is not supposed to be affected by the impact from the network or client-side behaviors.

Second, if $T_{net}$ is anomalous, but not any Server measure, then we consider it as an anomaly associated with the network. The underlying rationale is that because of the way we compute $T_{net}$ (§3), it is not intimately dependent on client behavior. This is also supported by the first observation above. Finally, we consider that the remaining cases are due to client behaviors (e.g., bot attacks).

### 5.2.2 Results

Using the diagnosis method described above, we determine the root cause for each of the detected anomalies and compare them with what is inferred and logged by the operators in the ticket database. We find that our results are perfectly consistent with the ticket database for those anomalies that are common to both methods.

Figure 19 shows three example anomalies. For each, SRT uses (blue) crosses and the left $y$-axis, another metric of interest uses (green) dots and the right $y$-axis, and the anomalous periods are marked using (red) circles. In the case of Figure 19(a), our tool diagnosed that the backend data center was the likely culprit because $T_{sc}$ and SRT were anomalous at the same time. Figure 19(b) was diagnosed as a network failure (due to a re-configuration of the routing weights between the CDN edge servers and the backend data centers) using our tool, as it was accompanied by anomalous $T_{net}$. On the other hand, the culprit in Figure 19(c) was the change of query richness during a major holiday event. Although no action needs to be taken to resolve this performance degradation, our tool helped the operators eliminate the possibilities of failures in the data centers or the network.

Across all SRT anomalies that our tool detected, we find that the fractions of the anomalies that were attributed to the wide-area network, data center, and client behaviors are 37%, 27%, and 36%, respectively. That the culprits are almost evenly distributed (with the data center being slightly lower), means that there is no silver bullet to significantly reduce the bulk of the SRT anomalies. The provider must work on reducing the impact of failures and unexpected behaviors on all three fronts. Perhaps the most surprising aspect is that client behaviors account for a third of the anomalies. Most of these are attacks on search infrastructure, but some are also caused by sudden changes in query richness (e.g., following a major newsworthy event).

# 6. IMPLICATIONS AND DISCUSSION

Our work has several implications for managing, understanding, and diagnosing the performance of large-scale Web services. We discuss a few of them in this section.

**Performance management:** A key goal of many Web services is to provide consistently good performance to their users. Work on this front has mainly focused on consistent request processing delays within the data center [6, 10]. Our findings, however, show that focusing on server processing time alone is insufficient and much of the variation in users' performance stems from factors such as network paths, browsers, and query types. While these factors are not under direct control of the service provider, there are ways in which their impact can be minimized.

For network paths, the service provider can send simpler pages for queries that come from clients behind slow last mile links. Using a data source that is different from the one used in this paper we find that the middle mile between the data center and the edge exhibits significant performance variations as well. This variations can be controlled through dedicated capacity between these points. One major provider appears headed in that direction already [23].

To ensure consistently fast SRTs across browsers, the providers can customize the responses and scripts to account for the strengths and weaknesses of individual browsers. Our data indicates that customization for the top three or four browsers would cover the vast majority of the queries.

To ensure fast SRTs for rich queries, the CDN edge servers can help reduce the burden on the user's end (browser) by simplifying dynamically the layout of the page and preprocessing some of the scripts. They can also cache more of the CSS, images, and javascript. Caching on the edge is not used heavily today because bits of the response pages are personalized. Small amount of computational capabilities on the edge servers (for generating personalized portions of the page) and a different layout of the response page can improve the amount of the content that users can fetch from the edge servers.

**Performance monitoring:** Our work highlights the difficulty of reasoning about service performance when it is measured across all users because such a measure is tainted by systemic changes in behaviors and characteristics of user population. This holds not only for search but for other Web services as well. For instance, Zander et al. [34] observed different fraction of users with IPv6 capabilities on weekends vs. weekdays for many services. (They could not fully explain this variation. Our results confirm their suspicion that it stems from residential and enterprise users having different characteristics.)

To provide greater insight into service performance, we are developing techniques to partition requests into equivalence classes that account for expected performance differences across network paths, browsers, and query types. This would enable the operators to better monitor and understand performance variation within a class, which otherwise gets masked by changes in the fractions of queries in individual classes.

**Performance diagnosis:** Another difficulty highlighted by our work is that of detecting and diagnosing performance anomalies. Past research has argued for using end-to-end performance metrics, rather than relying on low-level metrics such as server processing time, CPU or network utilization, because the anomalies in low-level measures may or may not cause anomalies in user experience [9, 20]. But our work shows that using such measures is challenging due to systemic variations. Prior work has looked at systems with roughly stationary response time characteristics because of a lack of significant diversity in query types, browsers, and network paths, as may be the case in enterprise networks [9, 20]. But in the Web services context, effective diagnosis of response time requires factoring out systemic variations and appropriately combining high- and low-level metrics.

# 7. RELATED WORK

Our work builds upon much prior work on the performance of Web services. One thread of work takes a client-side perspective to monitor the performance that a client experiences at various Web sites or to uncover factors that impact Web page load times. For example, WProf [32] and WebPagetest [4] are tools to measure the performance to any Web site and provide a timeline of relevant events that occur while the page is being loaded; and Butkietz et al. [11] examines the impact of page structure and server location of various Web sites on a client's performance. In contrast, our work takes a provider-side perspective, to uncover factors that impact the performance delivered by a Web service to all its active clients.

Another thread of work focuses on improving different aspects of Web services such as request processing, page design, and content delivery. For example, several works seek to make request processing predictable in a data center [7, 33]; tools exist to help developers follow the best practices for site layout and design [3, 5]; and many researchers have studied several facets of CDN design such as placement

and TCP behavior of edge servers [8, 12, 17, 19, 31] and proposed enhancements such as better redirection and caching, and hybrid CDNs [14, 16, 18, 21, 28]. In contrast to this body of work, we take an end-to-end view, which includes all relevant aspects, of the performance of a Web service. Further, instead of focusing on specific design enhancements, we seek to explain and diagnose performance variations given the design of a modern, large-scale service.

# 8. CONCLUSIONS

Our work addresses the challenges in understanding and diagnosing the performance of large-scale, Web services. We showed that the response time for a large service varies widely and, surprisingly, increases during off-peak hours when the query load is lower. We developed an analysis framework that reveals that this variation stems from systemic shifts in user characteristics—the source networks, the nature of queries, and the browsers used. In contrast, server processing times are relatively stable. We also developed and deployed a technique that detects and diagnoses service performance anomalies by factoring out the impact of such systemic variations. We found that our technique detected three times more anomalies than conventional methods that do not account for systemic variations.

## Acknowledgments

We thank Cheng Huang, Ritchie Hughes, Mujtaba Khambatti, Venkat Narayanan, Aditya Telidevara, for feedback in the initial stages of the project. We also thank Aditya Akella, Meg-Walraed Sullivan, the SIGCOMM reviewers, and our shepherd, Walter Willinger, for feedback on earlier drafts of this paper.

# 9. REFERENCES

[1] Central limit theorem. `http://en.wikipedia.org/wiki/Central_limit_theorem`.

[2] Measure page load time. `https://developers.google.com/speed/docs/pss/LatencyMeasure`.

[3] Pagespeed tools family. `https://developers.google.com/speed/pagespeed/`.

[4] WebPageTest - website performance and optimization test. `http://www.webpagetest.org/`.

[5] Yslow. `http://yslow.org/`.

[6] T. Abdelzaher, K. Shin, and N. Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 2002.

[7] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting Web search result preferences. In *ACM SIGIR*, 2006.

[8] M. Al-Fares, K. Elmeleegy, B. Reed, and I. Gashinsky. Overclocking the Yahoo! CDN for faster Web page loads. In *IMC*, 2011.

[9] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM*, 2007.

[10] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 2003.

[11] M. Butkiewicz, H. Madhyastha, and V. Sekar. Understanding website complexity: Measurements, metrics, and implications. In *IMC*, 2011.

[12] Y. Chen, S. Jain, V. Adhikari, and Z. Zhang. Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution. In *IMC*, 2011.

[13] J. Dean. Achieving rapid response times in large online services. In *Berkeley AMPLab Cloud Seminar*, 2012.

[14] M. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In *USENIX NSDI*, 2004.

[15] A. Gelman. Analysis of variance, 2005. `http://www.stat.columbia.edu/~gelman/research/unpublished/econanova.pdf`.

[16] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *IEEE INFOCOM*, 2005.

[17] C. Huang, A. Wang, J. Li, and K. Ross. Measuring and evaluating large-scale cdns. In *IMC*, 2008.

[18] H. Jiang, J. Li, Z. Li, and X. Bai. Performance evaluation of content distribution in hybrid CDN-P2P network. In *IEEE Future Generation Communication and Networking*, 2008.

[19] K. Johnson, J. Carr, M. Day, and M. Kaashoek. *The measured performance of content distribution networks*. Elsevier, 2001.

[20] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *ACM SIGCOMM*, 2009.

[21] J. Kangasharju, K. Ross, and J. Roberts. *Performance evaluation of redirection schemes in content distribution networks*. Elsevier, 2001.

[22] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the Web: Listen to your customers not to the HiPPO. In *SIGKDD*, 2007.

[23] R. Krishnan, H. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *IMC*, 2009.

[24] A. Pinto. How to measure page load time with google analytics. `http://www.yottaa.com/blog/bid/215491/How-to-Measure-Page-Load-Time-With-Google-Analytics`.

[25] D. G. Rees. *Foundations of statistics*, volume 214. Chapman & Hall, 1987.

[26] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *ACM IMW*, 2002.

[27] A. Singhal and M. Cutts. Official Google webmaster central blog: Using site speed in Web search ranking, 2009. `http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html`.

[28] Y. Song, V. Ramasubramanian, and E. Sirer. Optimal resource utilization in content distribution networks. Technical Report TR2005-2004, Cornell University, 2005.

[29] S. Stefanov. Progressive rendering via multiple flushes. `http://www.phpied.com/progressive-rendering-via-multiple-flushes/`.

[30] W. Taylor. Change-point analysis: a powerful new tool for detecting changes, 2000. `http://www.variation.com/cpa/tech/changepoint.html`.

[31] S. Triukose, Z. Wen, and M. Rabinovich. Measuring a commercial content delivery network. In *ACM WWW*, 2011.

[32] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystify page load performance with WProf. In *USENIX NSDI*, 2013.

[33] R. White and S. Dumais. Characterizing and predicting search engine switching behavior. In *ACM conference on Information and knowledge management*, 2009.

[34] S. Zander, L. Andrew, G. Armitage, G. Huston, and G. Michaelson. Mitigating sampling error when measuring Internet client IPv6 capabilities. In *IMC*, 2012.