An Analysis of Usage Locality
for Data-Centric Web Services
(TR-2005-866)

Congchun He
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

congchun@cs.nyu.edu

Vijay Karamcheti
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

vijayk@cs.nyu.edu

ABSTRACT

The growing popularity of XML Web Services is resulting in a significant increase in the proportion of Internet traffic that involves requests to and responses from Web Services. Unfortunately, web service responses, because they are generated dynamically, are considered "uncacheable" by traditional caching infrastructures. One way of remedying this situation is by developing alternative caching infrastructures, which improve performance using on-demand service replication, data offloading, and request redirection. These infrastructures benefit from two characteristics of web service traffic — (1) the open nature of the underlying protocols, SOAP, WSDL, UDDI, which results in service requests and responses adhering to a well-formatted, widely known structure; and (2) the observation that for a large number of currently deployed data-centric services, requests can be interpreted as structured accesses against a physical or virtual database — but require that there be sufficient locality in service usage to offset replication and redirection costs.

This paper investigates whether such locality does in fact exist in current web service workloads. We examine access logs from two large data-centric web service sites, SkyServer and TerraServer, to characterize workload locality across several dimensions: data space, network regions, and different time epochs. Our results show that both workloads exhibit a high degree of spatial and network locality: 10% of the client IP addresses in the SkyServer trace contribute to about 99.95% of the requests, and 99.94% of the requests in the TerraServer trace are directed towards regions that represent less than 10% of the overall data space accessible through the service. Our results point to the substantial opportunity for improving Web Services scalability by on-demand service replication.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network operations–Network management, Network monitoring; H.1.0 [**Models and Principles**]: General

General Terms

Measurement, Performance

Keywords

Web Services, Workload characterization, Locality, Service replication, Data offloading, Database partition

1. INTRODUCTION

The World Wide Web has gradually undergone a transformation from its primarily read-only, information-centric roots into an infrastructure that provides programmatic access to a variety of sophisticated services. This last trend is likely to become widespread because of the growing popularity of the XML Web Services architecture, which specifies a standard set of protocols (SOAP, UDDI, and WSDL) to enable dynamic discovery and interaction. Several of the largest services on the Internet today offer a web services portal, with representative examples including Microsoft's MapPoint Web Services, Amazon's Web Services, Google's Web APIs, and the TerraServer and SkyServer imagery servers.

Consequently, a significant fraction of Internet traffic in the not so distant future is likely to involve requests to and responses from web services. Unfortunately, because responses from Web Services are dynamically generated, such traffic will be considered "uncacheable" by traditional web caching infrastructures and hence see higher latencies and lower scalability. However, there are two characteristics of web service traffic that can, in principle, be exploited to remedy this situation. First, the open protocols underlying the web services architecture imply that service requests are "inspectable" being well-formatted (e.g., SOAP-based) and following a widely-known interface (e.g., published in WSDL). Second, for a large number of currently deployed data-centric services, it is possible to interpret requests as structured accesses against a (physical or virtual) service database; the request parameters point to the region of the database that is used to satisfy the request. As an example, one might think of the maps service provided on the MapPoint site[12], which takes as input the latitude and longitude of a location and a desired map scale, and returns a map which is dynamically composed from pre-computed image data around that location in the back-end database. Together, the two characteristics point to the potential of improving service performance using alternative caching infrastructures, which perform on-demand service replication, data offloading, and request redirection by inspecting requests to infer service usage patterns.

However, given the relatively high costs of service replication and/or request redirection, such infrastructures can achieve performance improvements only if there exists sufficient *locality* in service usage. For practicality, one would ideally like to see only a small subset of the overall service data or functionality being replicated, at a small number of locations in the network, and in a fashion that benefits requests over a long time duration. Thus, before one can proceed with building and deploying such infrastructures, one needs a better understanding of the workloads seen by such services to answer questions about whether sufficient locality exists and if so of what kind.

In this paper, we investigate the characteristics of workloads on two large-sized web service sites, SkyServer [16] and TerraServer[2], whose usage logs we could obtain access to. The SkyServer site presents data from the Sloan Digital Sky Survey, and its usage logs are publicly accessible. The TerraServer site provides a front-end to one of the world's largest online databases comprising map and relief data from the US Geological Survey (USGS)[3]. We were only able to obtain a 1-day trace for this site, albeit one involving 24 M requests. Although the results of our study are obtained on these specific traces, it is our belief that similar characteristics apply to other data-centric services as well including those from MapPoint, Amazon, and Google.

We examine the web service requests to quantitatively characterize how much *temporal*, *spatial*, and *network* locality they exhibit. Temporal locality refers to the property that within a short time period, a subset of objects of the database are more frequently accessed than others; spatial locality to the property that a subset of objects of the database are more frequently accessed than the others; and network locality to the property that such requests originate from a small group of users that are colocated in network space. Understanding these characteristics helps drive the design of the service replication infrastructures.

Our study had to address several challenges: unifying the multiple representations of service requests, grouping the large population of users, and modelling accesses to a large-sized multi-attribute database. For the first two of these issues we adopted solutions that are natural: using the WSDL specification of the service interfaces and a data-cleaning procedure to aggregate related requests, and using client IP addresses to loosely group users into 8-bit or 16-bit sub-networks. The third issue above is harder to deal with both because the relational schema of the database underlying a production service is difficult to obtain in practice, and even when disclosed, would require a complicated data model to efficiently represent and analyze accesses, at different granularities, against a huge volume of data in a multidimensional data space. In this paper, we discuss the design of a *logical view* of the service database, which relies on information from the service interfaces to present a simpler, analysis-friendly view of accesses to the original database.

Our results verify that there exist ample opportunities for improving web service scalability using on-demand service replication. Specifically, we find that:

- Both workloads exhibit a high degree of spatial and network locality. As examples, 10% of the client IP addresses in the SkyServer trace contribute to about 99.95% of all requests over a 4-month period, and 99.94% of the requests in the TerraServer trace are directed towards regions that represent less than 10%

of the overall data space accessible through the service.

- Such locality is present across multiple data region granularities, network levels, and time scales, affording considerable flexibility for the service replication infrastructures to choose an appropriate operating point. For example, in the SkyServer trace, when client addresses are aggregated into groups identified by the first two octets of the address, 10% of the groups still contribute to nearly 80% of all requests. Moreover, this behavior is present even at the timescale of a week.

The rest of the paper is organized as follows. Section 2 briefly introduces the SkyServer and TerraServer web sites and provides details about the access traces. Section 3 presents a taxonomy for the kinds of request locality that our study characterizes, and introduces a logical view of web service databases to simplify workload analysis. The analysis methodology and our findings are presented in Sections 4 and 5 respectively. Finally, we discuss related efforts and possible extensions to our work (Section 6) and conclude.

2. BACKGROUND

To assess whether usage of real-world data-centric web services does in fact exhibit the kinds of locality required for service replication techniques to be beneficial, we analyzed request logs from two such sites, SkyServer and TerraServer. Our choice of these sites was driven by the fact that we were able to obtain access to their request logs, often the major impediment to performing a study similar to ours. SkyServer's web logs are publicly available.

The SkyServer web site provides Internet access to the public Sloan Digital Sky Survey (SDSS) data for both astronomers and for science education. The SDSS is a 5-year survey of the Northern sky (10,000 square degrees) to about 1/2 arcsecond resolution using a modern ground-based telescope. Its goal is to characterize about 200 M objects in 5 optical bands, and measure the spectra of a million objects[4]. The web site has been operating since June 5, 2001 and receives approximately 3M requests every month.

The TerraServer web site is one of the world's largest online databases, providing free public access to a vast data store obtained from the US Geological Survey (USGS). The web site contains 3.3 TB of high resolution USGS aerial imagery and USGS topographic maps. The TerraServer web site has been operating since June 1998 and is heavily used: a typical day sees 25M - 30M requests.

The underlying architecture of both sites is similar and built on top of Microsoft's IIS and .NET framework offerings: a front-end IIS web server accepts HTTP requests and passes them to corresponding HTTP handlers (e.g., an ASP, ASPX, or ASHX handlers) for processing. The HTTP handlers, which implement the service functionality, query against a back-end database server to generate responses by formatting the returned records into HTML pages/SOAP messages. The access logs we work with are recorded by the IIS server, and include requests for static content; we filter out such requests from our analysis as described below. Because of its heavier load, the TerraServer site uses a web server farm, with each server logging its own trace. Our analysis is on an aggregate trace obtained by merging these individual traces in timestamp order.

2.1 Trace Structure

Table 1: Entry structure in the two web traces.

Index	Field	Index	Field
1 − 3	**yy, mm, dd**	1	**date**
4 − 6	**hh, mi, ss**	2	**time**
7	seq	3	s-computername
8	logID	4	cs-method
9	**clientIP**	5	**cs-uri-stem**
10	op	6	**cs-uri-query**
11	**command**	7	cs-username
12	error	8	**c-ip**
⋮	⋮	⋮	⋮

(a) SkyServer Weblog (b) TerraServer Weblog

Our analysis of the SkyServer site is based on a four month trace from January 1, 2004 to April 30, 2004, containing 11.4 M requests. For the TerraServer site, we could only obtain access to a single day's trace, April 5, 2004, which contained 24 M requests. Table 1 shows the entry structure of request logs at the two web sites. Each entry in the request log contains the following information: a timestamp, an IP address of the client that made the request, the name of the requested page/service, and the supplied parameters for the request. For the SkyServer request log, the first 6 fields ("yy", "mm", "dd", "hh", "mi" and "ss") together provide the timestamp information; field "clientIP" provides the client IP address information; field "command" provides both the name of the requested page/service and the supplied parameters for the request. For the TerraServer request log, the "date" and "time" fields together provide the timestamp information; field "c-ip" provides the client IP address information; field "cs-uri-stem" provides information on the name of the requested page/service; and field "cs-uri-query" provides information about parameters supplied with the request.

The logs consist of requests for both static and dynamic content. Our analysis focuses on the latter requests, which involve accesses to a back-end database. We apply a broad interpretation for such requests, including both services that are invoked using SOAP, as well as ASP .NET web applications that are invoked using HTTP GET/POST. Table 2 summarizes the fraction of each category of request.

For each dynamic requests, we parse the logs to identify the request command ("service") and parameters, discarding requests that were either ill-formatted in syntax or incompatible in semantics (this constituted a very small fraction of all requests, fewer than 0.34%). While we looked at a broader set of requests, here we restrict our attention to the most frequently invoked services (out of thousands) that dominate the dynamic traffic at each site. Table 3 summarizes the statistics of each kind of request. The client IP address columns correspond to the number of unique addresses requesting that service, and unique groups of addresses that share either their first three octets (corresponding to a 8-bit subnet) or their first two octets (loosely corresponding to a 16-bit subnet).

Services for the SkyServer site include:

- **x_rect.asp**: which takes as input a rectangle in the sky (specified by the rectangle center, width, and height expressed in terms of the sky coordinates of right ascension and declination) and returns a list of objects found in that rectangle.

- **x_radial.asp**: which takes as input a circle in the sky (specified by a center point and a radius expressed in terms of the sky coordinates of right ascension and declination) and five optional optical bands, and returns a list of objects found in that circle.

- **getjpeg.aspx**: which takes as input a rectangle in the sky, a desired map scale and a set of drawing options, and returns a JPEG image.

- **shownearest.asp**: which takes as input a circle in the sky, a desired map scale and a set of drawing options, and returns a thumbnail JPEG image.

- **x_sql.asp**: which takes as input an arbitrary SQL query against the back-end database, and returns the results in required format (HTML, XML or CSV).

For the TerraServer site, one request type dominates:

- **tile.ashx**: which returns a 200 pixels × 200 pixels imagery of the "Photo", "Topo", or "Relief" type corresponding to a point on the Earth's surface specified using the UTM coordinate system [1], on a desired map scale. Requests for the "Photo" type contribute to ∼ 85% of the whole, so we limit our attention to that type.

We characterize locality properties for each of these services, except *x_sql.asp*, whose requests make arbitrary queries again the underlying database. As reported in [4], the relational schema of the SkyServer database is very complicated, thereby making it difficult to analyze data region accesses by requests without additional information about the database internals.

3. LOCALITY IN SERVICE REQUESTS

Service replication infrastructures of the kind described in Section 1 need to make decisions about what to replicate/offload, where to locate service replicas, and when to perform such actions. To help answer these questions, our analysis focuses on quantitatively characterizing the presence of *temporal*, *spatial*, and *network* locality in the request logs.

Temporal locality refers to the property that within a certain time period, a subset of objects of the service database are more frequently accessed than others. Such locality arises from the presence of "hot" objects in the database whose popularity changes with time; for example, in the SkyServer site, such locality may be correlated with unusual astronomical phenomena such as a meteor shower. Detection of such locality can help alleviate hotspots and flash crowd-like effects at the service.

Spatial locality refers to the property that certain regions of the database are more frequently accessed than others. For example, certain book categories at an online bookstore like Amazon.com might receive a dominant fraction of all requests. Capturing spatial locality can help define appropriate "materialized views" to drive the database partitioning and replication process.

Network Locality refers to the property that a certain group of users, colocated in network space, are responsible for a dominant fraction of all requests to the service. Such

Table 2: Fraction of static and dynamic content.

Content	hits (M)	%	Content	hits (M)	%
SkyServer			**TerraServer**		
all	*11.402*	*100*	*all*	*24.220*	*100*
static	6.181	54.21	static	5.809	23.99
dynamic	5.221	45.79	dynamic	18.411	76.01

Table 3: Types of services showing request and client IP statistics.

Service	Requests		Client IPs		
	(M)	%	uniq.	8-bit	16-bit
SkyServer					
all	*5.221*	*100*	-	-	-
x_rect.asp	2.490	47.69	219	188	168
x_radial.asp	0.417	7.99	735	548	432
getjpeg.aspx	0.476	9.12	7410	5223	2860
shownearest.asp	0.339	6.49	9003	7153	3648
x_sql.asp	0.506	9.69	-	-	-
TerraServer					
all	*18.411*	*100*	-	-	-
tile.ashx	16.516	89.71	29029	25699	5466

Table 4: Logical views of services.

Service	Input	View
x_rect.asp	rectangle(in sky) 5 optical bands	2D: rectangle center-point
x_radial.asp	circle 5 optical bands	2D: circle center-point
getjpeg.aspx	rectangle, scale drawing options	3D: rectangle center-point, scale
shownearest.asp	circle, scale drawing options	3D: circle center-point, scale
tile.ashx	scale, point (in UTM) photo type	4D: scale, 3D point

locality typically arises because of shared interests among a group of users, e.g., one expects map services such Map-Point.com to exhibit significant geographical locality. Capturing network locality can help identify appropriate locations in the network to place service replicas.

To enable characterization of these three kinds of locality, we correlate the parameters of each service request with the region of the underlying database used to respond to the request. Ideally, we would like this correlation to be in terms of the physical back-end database. However, for most production services like SkyServer or TerraServer, obtaining the relational schema of the back-end database not only raises difficulties in practice, but even when available may prove overly cumbersome for detailed analysis. As an alternative, for data-centric web services that are our primary interest, we use the information provided in the documentation of the service interface to construct a simpler *logical view* of the database.

3.1 Logical view of the back-end database

The logical view of a service's database is constructed by examining the supplied parameters in the service's WSDL interface, and defining a virtual dataspace accessed by these parameters, assuming that they are numeric or alphabetic rangeable. Note that the logical view only approximates usage of the underlying physical database, not its structure.

We are often interested in only a subset of this overall dataspace corresponding to parameters that exhibit the most variation across requests. Thus, a view defines a multidimensional data space based on one or more parameter value ranges. Following the concept of *semantic regions* in database caching [10], we can partition such a data space into multiple disjointed regions at some granularity, and model service accesses at the level of these regions.

Our analysis makes use of the logical service views as follows. An individual service request can now be interpreted as an *edge* originating from a particular client IP address and directed to a point in such a multidimensional data space. The reason it is a point (instead of more generally being a region) is because to ensure efficiency, most services usually enforce a range limit on an individual request. As an example, on the SkyServer site, the *x_rect.asp* service only allows a request to search objects within a rectangle which has a maximum size of 0.2 degrees by 0.2 degrees in sky coordinates. By examining clustering of request edges in both the IP address space and the logical view data space, we can quantify the extent to which a request trace exhibits different kinds of locality.

Views for SkyServer and TerraServer services We take service *x_rect.asp* as an example to show how to construct a logical "view" for a service. As introduced in Section 2, the input for the service consists of 9 parameters, and hence its logical view might consist of 9 attributes: the coordinates of the center-point, width and height of a rectangle in the sky, and the optional 5 optical bands. An alternative view could consist of the center-point and the rectangle dimensions by clustering together the dimensions corresponding to the optional 5 optical bands; service requests in our traces exhibit relatively little variation for these values. Further simplification is possible by observing that each request can refer to a maximum rectangle of a known size, hence it suffices to define the view in terms of the coordinates of the rectangle's center-point.

Table 4 lists the logical views that we constructed for each of the five services. Notice that while the first 4 services have a rather simple view which consists of 2 or 3 dimensions, the last service's consists of 4 dimensions. In the rest of the paper, we use the following abbreviations to refer to these dimensions: for the SkyServer services, the map scale dimension is denoted as "S", the right ascension coordinate dimension is denoted as "X", and the declination coordinate dimension is denoted as "Y"; for the TerraServer service, the map scale is denoted as "S", the scene dimension (the "zone" coordinate in UTM) is denoted as "T", and the "easting" and "northing" dimensions are denoted as "X" and "Y", respectively. TerraServer uses these four parameters to identify a point on the earth's surface using the UTM system.

The parameters of each dynamic request in the traces described in Section 2 are translated to a point in the data space of the logical view for that service. In cases where two services share the same logical view (this happens in the case of *x_rect.asp* and *x_radial.asp* in our case), we pool

the requests into a combined group ordering them using the associated timestamp information.

4. METHODOLOGY

As stated earlier, our analysis goals are to characterize to what extent the request logs exhibit temporal, spatial, and network locality. Note however, that the locality structure depends on various parameters: the granularity of data space regions, how addresses are grouped, and the timescales of interest. To systematically examine the effect of these factors, we parameterize the request workload: a particular assignment of parameter values divides up the logical data space into partitions of a certain size, the trace into different time epochs, and the client IP addresses into different address groups.

We then examine locality patterns over a range of values for these parameters, as shown in Table 5. The role of the "IP address groups" and "timescale" parameters should be clear: we consider two groupings of client IP addresses, based on whether they share their first 2 or first 3 octets,[1] and different time epoch sizes ranging from an hour to the entire duration of the trace. The "partitioning policy" parameter controls along which dimension(s) and at what granularity the logical data space of the service is partitioned. For example, in the SkyServer *getjpeg.aspx* service, we examine 21 partitioning policies corresponding to seven choices of partition-by-dimension (`S, X, Y, SX, SY, XY, SXY`) , and three granularities for each of these choices (a granularity value of x implies that regions are defined by dividing each dimension of the partition into 2^x equal intervals). To explain the notation, a partitioning policy value of `XY10` corresponds to 2^{20} data space regions; each region spans $1/2^{10}$ of the range of the `X` and `Y` dimensions. For the TerraServer service *tile.ashx*, the granularity of each dimension to be partitioned is set to a fixed value: 11 regions for the "S" dimension, 10 for "T", and 40 for both "X" and "Y".

The *Cell* structure To efficiently analyze the request traces for a particular partitioning policy, we use a dynamic data structure, called *Cell*. A Cell basically defines a region in the data space (correspondingly, a partition of the database) and is responsible for collecting the requests that hit in this region. A Cell can recursively split itself along one or more dimensions of the data space to form a Cell tree, assuming each dimension is alphabetic or numeric rangeable. By specifying the ways to split the *Cell* structure and the depth of Cell tree, we can control how and at what granularity a data space is partitioned: the leaf nodes in a Cell tree represent a database partition at a certain granularity (Figure 1).

4.1 Characterizing locality

To reason about the three kinds of locality in a unified fashion, we compute a "load fraction" graph. Each point in this graph corresponds to the *maximum* request load (as a fraction of all requests) that is observed between a set of address groups and a set of data space partitions, under the

[1]The intention here was to group network addresses to ascertain if locality exists at higher levels of the network, say at the granularity of 8-bit and 16-bit subnets. We realize that the 2-octet grouping is only an approximation for the 16-bit subnet evaluation; a more precise analysis would require inferring the IP address-AS associations.

Table 5: Range of parameter values over which locality characteristics were investigated.

| Parameter	Values investigated
x_rect.asp	
partitioning policy IP address group timescale	$2^{\{X,Y\}}\backslash\phi \times \{4,7,10\}$ share first $\{2 \mid 3\}$ octets {hour, day, weekday, weekend, week, month, all}
getjpeg.aspx, shownearest.asp	
partitioning policy IP address group timescale	$2^{\{S,X,Y\}}\backslash\phi \times \{4,7,10\}$ share first $\{2 \mid 3\}$ octets {hour, day, weekday, weekend, week, month, all}
tile.ashx	
partitioning policy IP address group timescale	$2^{\{T,S,X,Y\}}\backslash\phi$ share first $\{2 \mid 3\}$ octets {hour, day (all)}
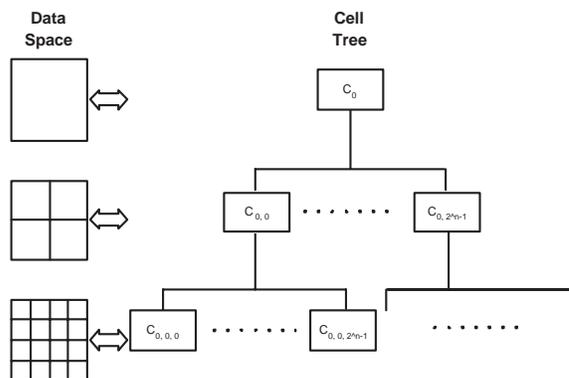
Figure 1: Data space partitioning vs. Cell splitting.

constraint that the cardinality of these two sets is bounded to certain values. Informally, the graph shows the potential benefits of caching a subset of data space view regions at a subset of the network locations from where requests originate. For example, in figure 3, a point at location (30, 10, 0.84) can be interpreted as meaning that "10% of 8-bit IP address groups contribute at most 84% of requests on 30% of the regions".

The computation of the maximum request load is NP-hard, so we use a greedy heuristic (Figure 2) to approximate the maximum request load. The heuristic takes as input two budgets (the thresholds on the fraction of IP address groups and of regions), a set of client IP address groups, a set of regions (represented as leaf nodes in the Cell tree) and a set of edges between the two sets where an edge represents one of the addresses in the address group making a request to the stated region; the edge weight denotes the number of such requests. The algorithm then "greedily" selects the maximal weight edges until the two budgets are exhausted. Clearly, the heuristic provides a lower bound on the achievable load fraction.

5. ANALYSIS

Inputs:
    C: a set of client IP addresses
    L: a set of leafs in Cell-tree
    E: a set of edges between C and L where:
        $e_{i,j} \in E \Leftrightarrow$ client c_i has requests hit on leaf Cell l_j;
        the weight of $e_{i,j}$ is the number of requests
    B_c, B_l: two budgets

Variables:
    C': a set containing the selected elements from C;
        $|C'| \le B_c$
    L': a set containing the selected elements from L;
        $|L'| \le B_l$

Output:
    the total weight of edges between C' and L'

Algorithm:
    – sort the edges in decreasing weight order
    – select the heaviest edges until one of the budgets is exhausted (edge $e_{i,j}$ is selected $\Rightarrow C' = C' \cup \{c_i\}$ and $L' = L' \cup \{l_j\}$)
    – once a budget is exhausted, select the heaviest edges that originate from the nodes in that selected set until the other budget is also exhausted
    – select all remaining edges whose head and tail are already in C' and L'

Figure 2: Greedy computation of "load fraction".

Figure 3: IP_3_PP_XY10_TS_All: Spatial locality for requests accessing the *x_rect.asp* service in the SkyServer trace.

Our results show that there exists high spatial and network locality in both workloads on the SkyServer and TerraServer sites. In this section, our discussion focuses on *x_rect.asp* and *tile.ashx*, the two most representative services at each site.

Each graph presented in this section is indexed with a string of the form "IP_*n*_PP_*dims*[*l*]_TS_*epoch* [_*val*]", referring to a specific assignment of parameters (IP address grouping, partitioning policy and timescale) as discussed in Table 5; "val" indicates that the value is computed as an avg/max/min of request loads from multiple time epochs.

Overall locality characteristics
For SkyServer's *x_rect.asp* service, corresponding to finest granularity regions (where the data space is divided into 2^{20} regions), addresses grouped into 8-bit subnets, and the largest timescale (4 months spanning the entire trace), Figure 3 shows that: (1) 10% of client IP addresses contribute about 99.95% of requests; (2) 84.04% of the requests hit on 30% of regions in the data space; and (3) because of (1), for any specific certain fraction of regions, increasing the fraction of IP addresses from 10% to 100% does not affect the computed load fraction too much (the maximum load fraction that can be added is only 0.05%).

Similarly, for TerraServer's *tile.ashx* service, corresponding to finest granularity regions (where the data space is divided into 176,000 regions), addresses grouped into 8-bit subnets, and the largest timescale (1 day, spanning the entire trace), Figure 4(a) shows that: (1) 10% of client IP addresses contribute about 83.94% of requests; (2) 99.94% of requests hit on 10% of regions in the data space; and (3)
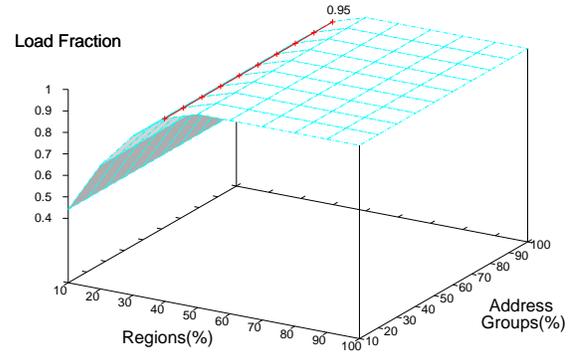
because of (2), for any specific certain fraction of IP addresses, increasing the fraction of regions from 10% to 100% does not affect the computed load fraction too much.

The results imply that for both workloads, caching a small fraction of regions at a small fraction of subnets could potentially reduce a large fraction of service traffic in the network. The results in Figure 4(a) indicate that if we could only cache up to 10% of the regions at up to 10% of the subnets, we could still cover (i.e., potentially improve the performance of) as much as 83% of the overall request traffic. Zooming in on this figure (Figure 4(b)) reveals that even if we could only cache at most 1% of the database view regions at no more than 1% of the subnets, we could still cover as much as 63% of the overall request load.

Figures 5 and 6 present the detailed characteristics of workload locality for the *x_rect.asp* and *tile.ashx* services, over a range of representative values for data space region granularity, addresses grouping and time epoch size. We discuss the salient points of these graphs below.

Impact of IP address grouping
To understand if the locality structure is different for higher levels of the network (since any practical replication strategy would need to share a replica among multiple clients), we examined how the load fraction graph changes when all IP addresses that share either their first two octets (loosely corresponding to a 16-bit subnet) or their first three octets (corresponding to a 8-bit subnet) were pooled together into an address group. In the *x_rect.asp* trace, we found that for both grouping scenarios, 10% of the groups contribute to about 99.95% of the requests, suggesting that a large number of requests continue to benefit even if replicas can only be created at higher network levels (Figures 5(c) and (f)). A similar trend was observed for the *tile.ashx* trace. The locality structure stayed the same for the finer grouping, and only reduced slightly for the coarser one where 10% of the groups ended up contributing to 79.2% of the overall requests (Figures 6(a) and (b)).

The observations imply that a service replication infrastructure has considerable flexibility in choosing how to improve service performance, and can strike an appropriate tradeoff between prioritizing client-perceived latency and data offloading costs (which would be lower with fewer repli-
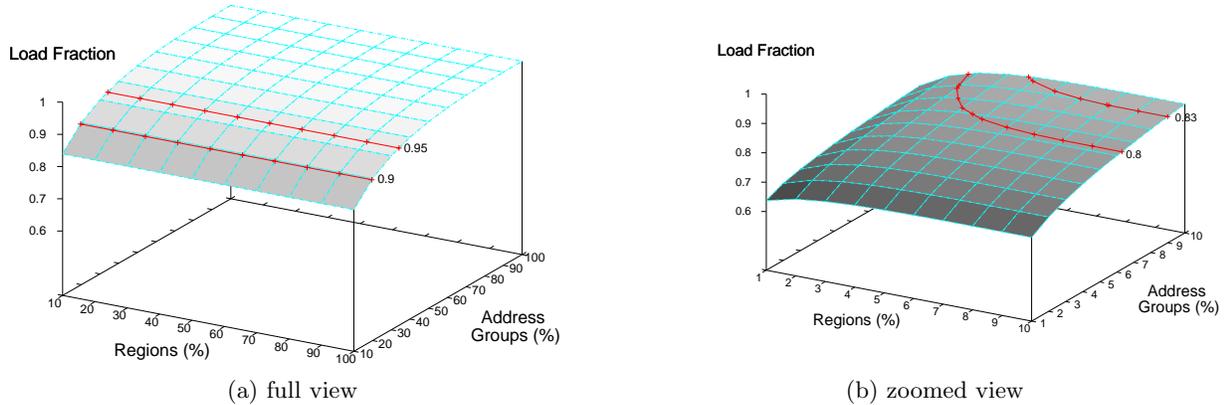
(a) full view

(b) zoomed view

Figure 4: IP_3_PP_TSXY_TS_All: Spatial locality for requests accessing the *tile.ashx* service in the TerraServer trace.

cas at higher levels of the network).

Impact of region granularity

To understand how much of the locality structure is still present for coarser granularity regions in the data space (again, for practical reasons, one would like to replicate large contiguous blocks of the data space as opposed to individual scattered relations), we examined the nature of the load fraction graphs for different granularity values. We find that significant amounts of locality continues to be present even at coarser granularities.

In the *x_rect.asp* trace, when client IP adddresses are grouped into 8-bit subnets, one still sees a large fraction of client requests targeting a small number of coarser-grained regions. Compared to 93.66% of requests hitting 30% of the regions, when region size is $1/2^{10}$-th of the data space size along each dimension, 88.42% requests hit on 30% of the regions when the region size grows by a factor of 8 along each dimension, and 86.22% of requests continue to hit on the same fraction of regions even when this factor goes up to 64 (Figures 5(f), (g) and (h)).

The figures also show that at the finest region granularity, the network locality measured is lower than at coarser granularity when the fraction of address groups is low (10%): 60.67% of requests hitting on 30% of regions, as opposed to 88.38% and 86.19% in the other two figures. This behavior is an artifact of our greedy algorithm, which because it does not prioritize one budget over another, may occasionally mis-select an edge whose weight may be maximal, but whose client IP address group contributes fewer hits than another (e.g., because the second address might be involved in multiple edges).

More interestingly, our analysis reveals different locality behaviors for different region shapes suggesting that, for a given region size, locality can be optimized with additional service-specific knowledge. For example, Figures 5(a) and (b) indicate that compared to 54.73% of requests hitting 30% of the regions, when the logical data view is partitioned along the right ascension coordinate (X), 77.84% of requests hit on 30% of the regions when the view is partitioned along the declination coordinate (Y). A similar observation also

holds for the *tile.ashx* trace (Figures 6(c) and (d)).

Impact of time epoch size

To understand whether the locality structures develop over long timescales or are even present over short timescales, we examined how the load fraction graph varies with different time epoch sizes. The *x_rect.asp* trace, which contains few requests for shorter time epochs, shows larger variations in locality patterns for smaller time epochs than the *tile.ashx* trace. For the former, Figures 5(d), (e) and (f) show that 10% of the address groups contribute only 8.42% of requests for a daily timescale while this fraction goes up to 99.63% for a weekly timescale and 99.95% for a monthly timescale. For TerraServer's *tile.ashx* service, 10% of the address groups continue to contribute to at least 80% of the overall requests even for hour-long epochs (Figures 6(b) and (g)).

The observations imply that locality is in fact affected by the size of the time epoch, with additional service-specific information possibly being required to guide a service replication infrastructure in its choice of an appropriate epoch size at which to detect and optimize locality.

Deviation of measured locality

To understand the variations in locality over different time epochs, we examined three kinds of statistics for the load fraction, the minimum, average and maximum values, over the entire trace duration. Contributing to these statistics were the load fractions measured at each time epoch. The results indicate that the *x_rect.asp* trace exhibits larger deviations of load fraction compared with the *tile.ashx* trace. This observation likely results from the smaller number of requests seen over smaller time epochs in the SkyServer trace.

6. RELATED WORK AND DISCUSSION

The work presented in this paper has focused on characterizing web service requests in terms of how much locality they exhibit with respect to the data regions of the backend database involved in generating their responses. Our work is therefore related to previous efforts that have characterized different aspects of web traffic.

Pitkow [14] has presented a survey of web characterization studies and there are a number of other detailed studies that have examined web workloads [13, 5, 6, 11, 7, 17]. However,
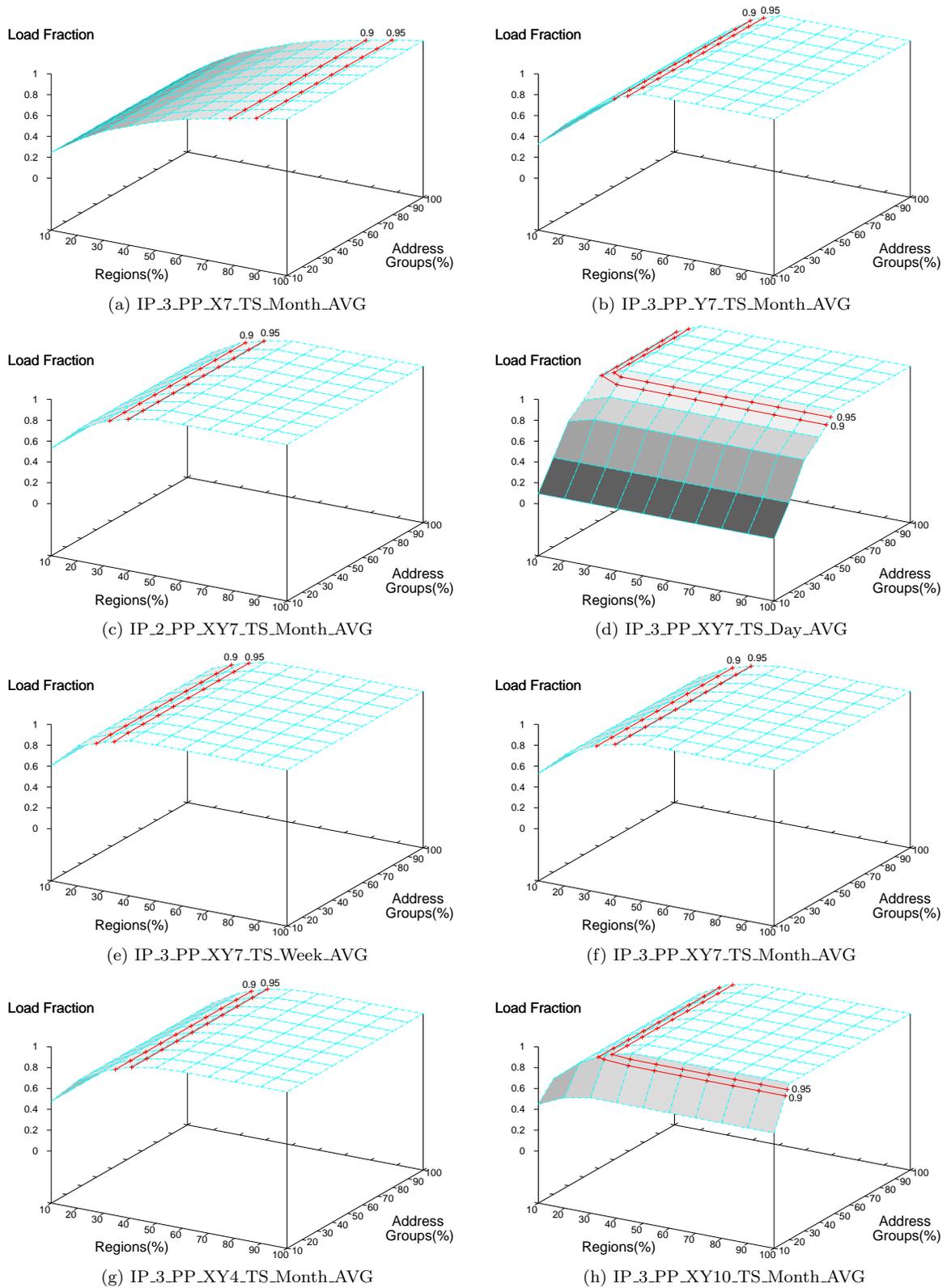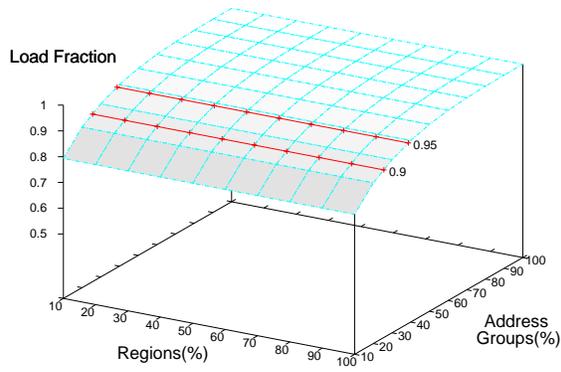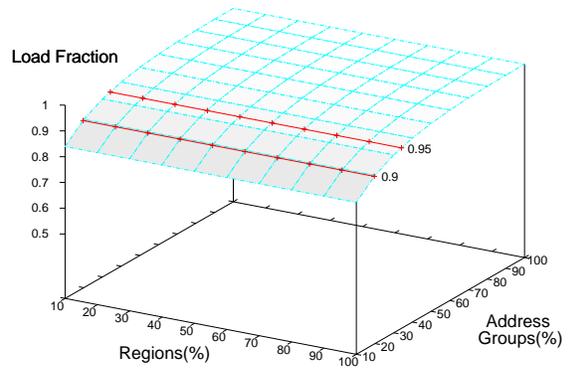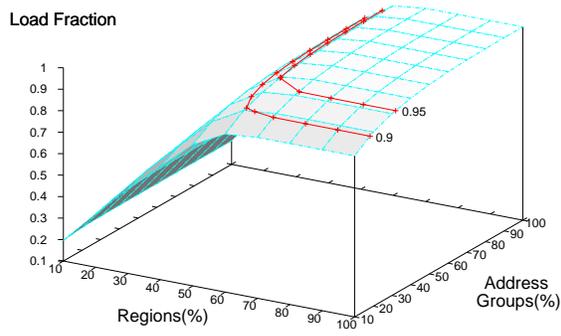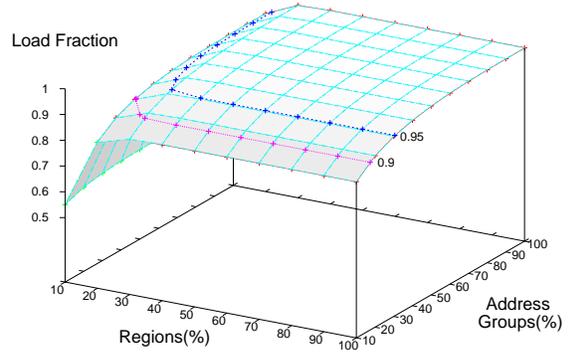
Figure 5: Workload locality for SkyServer's *x_rect.asp* service.
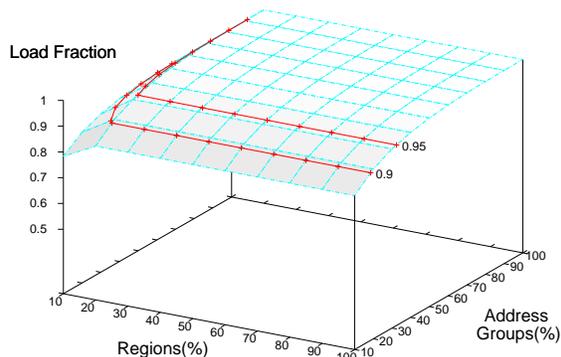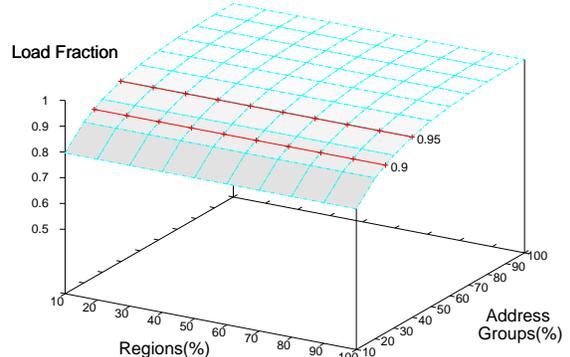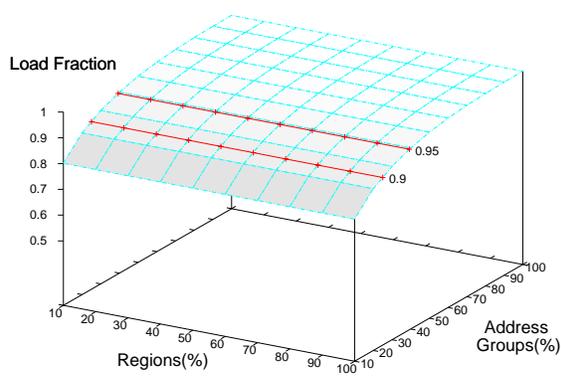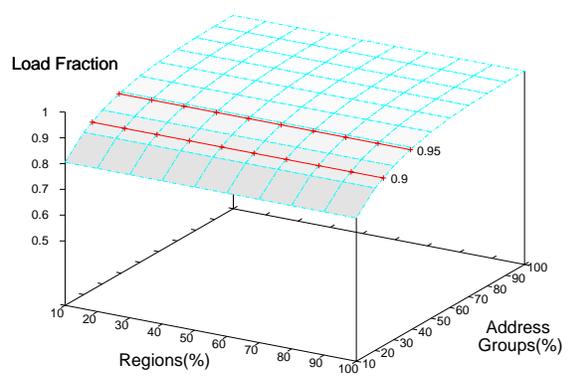
Figure 6: Workload locality for TerraServer's *tile.ashx* service.

most of these works have focused on characteristics of static web content, and their results do not directly apply to the case of dynamic web content. Relatively few studies [18, 8, 9, 15] have examined the characteristics of dynamic content: interestingly these studies have verified both the need for and likely benefit from caching content at sub-document granularity and moving content generation to the network edge; both these ideas can be thought of as a special case of service replication infrastructures that provide the motivation for this work.

Our work extends the above studies to the context of data-centric web services. Given this context, our definitions of temporal, spatial, and network locality differ somewhat from those used in previous studies [5, 11, 7, 17], but retain a similar spirit. From the methodology viewpoint, most notable about our approach is its use of logical views to model accesses against the back-end database and the employment of load fraction graphs as a means to quantitatively describe different kinds of locality. Logical views were influenced by the notion of semantic regions [10] from the database caching literature, where they refer to a range of relation values accessed by a group of queries (requests). The difference in our work is that such regions may only be *virtual*, serving to specify the internal service data required for servicing a group of requests.

The positive findings of locality across multiple dimensions points to the benefits for web service access that are likely from an alternative caching infrastructure that inspects service requests to perform on-demand service replication and request redirection. We are currently developing such an infrastructure. While this study serves as a starting point, additional work is required to further direct the choices in such infrastructures:

- Our conclusions in this study need to be validated by examining traces from other services and that span longer durations than we were able to obtain access to. It would also be interesting to examine if our methodology can be applied to other data-intensive services (e.g., Amazon's online book store services) or even to computation-intensive services. The critical issue here is of course obtaining access to traces for conducting the analysis.

- A shortcoming of our work is its inability to characterize the cost of offloading request handling for a particular data region to a service replica. Our methodology only characterizes the relative sizes of regions as opposed to the absolute costs. The latter information can be inferred by correlating the backend data schema and other proprietary service-specific information with our findings. The latter does need to be done in practice: the number of services provided by a large-sized web site like SkyServer or TerraServer could be very large. Our methodology associates a different data space with each service. However, not surprisingly, some of the service logical views might share common attributes in the underlying database. Knowing this association can also help one reason about cross-service locality in addition to the intra-service locality we have focused on in this study.

7. CONCLUSION

In this paper, we have investigated the workload characterization of data-centric Web Services in terms of three kinds of locality: temporal, spatial, and network. Our analysis is based on the web traces from the SkyServer and the TerraServer sites and indicates that both workloads exhibit high spatial and network locality. Possible benefits from such locality include the possibility of designing alternate caching infrastructure for web services that relies on on-demand database view customization and replication at appropriate locations in the network.

REFERENCES

[1] The Universal Transverse Mercator projection and grid system. http://www.maptools.com/UsingUTM/.

[2] TerraServer.com. http://www.terraserver-usa.com/, 2003.

[3] The U.S. Geological Survey. http://www.usgs.gov/, 2004.

[4] A. Szalay and et al. The SDSS DR1 SkyServer: Public Access to a Terabyte of Astronomical Data. http://skyserver.sdss.org/dr2/en/skyserver/paper/.

[5] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveria. Characterizing reference locality in the www. In *Proc. of PDIS '96*, 1996.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. IEEE INFOCOM 1999*, pages 126–134, 1999.

[7] M. Busari and C. L. Williamson. On the sensitivity of web proxy cache performance to workload characteristics. In *Proc. IEEE INFOCOM 2001*, pages 1225–1234, 2001.

[8] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM'99)*, 1999.

[9] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. A publishing system for efficiently creating dynamic web content. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM'00)*, 2000.

[10] S. Dar, M. J. Franklin, B. Jonsson, D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. In *Proc. 22nd VLDB Conference*, pages 330–341, 1996.

[11] A. Mahanti, D. L. Eager, and C. L. Williamson. Temporal locality and its impact on web proxy cache performance. 42(2). Special Issue on Internet Performance Modelling.

[12] Microsoft Corporation. Microsoft MapPoint Web Services. http://www.microsoft.com/mappoint/default.mspx, 2003.

[13] J. C. Moful. Hinted caching in the web. In *Proc. of the 7th workshop on ACM SIGOPS European workshop: Systems support for worldwide applications*, pages 103–108, 1996.

[14] J. E. Pitkow. Summary of www characterizations. *World Wide Web*, 2:3–13, 1998.

[15] W. Shi, R. Wright, E. Collins, and V. Karamcheti. Workload characterization of a personlized web site - and its implications for dynamic content caching. In *Proc. of the 7th Intl. Conf. on Web Content Caching and Distribution (WCW'02)*, 2002.

[16] Sloan Digital Sky Servey. SkyServer Projects. `http://skyserver.sdss.org/`, 2003.

[17] Q. Wang, D. Makaroff, H. K. Edwards, and R. Thompson. Workload characterization for an e-commerce web site. In *Proc. of the 2003 conf. of the Centre for Advanced Studies conf. on Collaborative research*, pages 313–327, 2003.

[18] C. E. Wills and M. Mikhailov. Studying the impact of more complete server information on web. In *Proc. of the 7th Intl. Conf. on Web Content Caching and Distribution (WCW'00)*, 2000.