ECHO: Recreating Network Traffic Maps for Datacenters with Tens of Thousands of Servers

Christina Delimitrou[*], Sriram Sankar[†], Aman Kansal[‡] and Christos Kozyrakis[*]
[*]Electrical Engineering Department, Stanford University, {cdel, kozyraki}@stanford.edu
[†]Microsoft, srsankar@microsoft.com
[‡]Microsoft Research, kansal@microsoft.com

***Abstract*—Large-scale datacenters now host a large part of the world's data and computation, which makes their design a crucial architectural challenge. Datacenter (DC) applications, unlike traditional workloads, are dominated by user patterns that only emerge in the large-scale. This creates the need for concise, accurate and scalable analytical models that capture both their temporal and spatial features and can be used to create representative activity patterns. Unfortunately, previous work lacks the ability to track the complex patterns that are present in these applications, or scales poorly with the size of the system. In this work, we focus on the network aspect of datacenter workloads. We present ECHO, a scalable and accurate modeling scheme that uses hierarchical Markov Chains to capture the network activity of large-scale applications in time and space. ECHO can also use these models to re-create representative network traffic patterns. We validate the model against real DC-scale applications, such as Websearch and show marginal deviations between original and generated workloads. We verify that ECHO captures all the critical features of DC workloads, such as the locality of communication and burstiness and evaluate the granularity necessary for this. Finally we perform a detailed characterization of the network traffic for workloads in DCs of tens of thousands of servers over significant time frames.**

I. Introduction

As the world's computation continues to migrate into massive datacenter (DC) infrastructures, developing highly efficient systems for these computing platforms has become increasingly critical. DC architectures are still in their relative infancy and when optimizing for performance, efficiency, or cost of ownership (TCO), architects must take into account the unique characteristics that dominate the behavior of large-scale applications. Understanding this behavior requires detailed workload characterization and is crucial not only from the systems but from the data analytics perspective as well.

DC applications are radically different from conventional workloads in several ways; first, privacy concerns make their source code, user behavior patterns and datasets rarely publicly available. This seriously hinders accurate and convincing studies. Second, DC applications experience activity patterns that cannot be reproduced or approximated by traditional benchmarks in standardized experimental environments because they only emerge from user behavior in the large scale, such as localized hotspots. Third, the cost of deploying experimental system configurations in a production environment is prohibitive both from the time and the cost perspective. This increases the importance of concise, accurate and scalable models that representatively capture the behavior of large-scale workloads and can be used to create realistic access patterns.

The network component of DC applications reflects a large fraction of user patterns both in time and space. It is often responsible for Quality of Service (QoS) guarantees violations and accounts for a significant portion of the infrastructure's TCO. Despite the obvious merit in developing representative analytical models that capture DC network traffic, unfortunately previous work lacks the ability to reflect the complex spatial and temporal patterns that emerge in the large-scale. For a workload model to be useful in the context of large-scale DCs it needs to have three main properties: (a) *accuracy*, so that the information in the model closely resembles the actual behavior of the application, (b) *modularity*, so that it can: (i) adjust the granularity of information to the needs of an application and (ii) be reconfigurable and compatible with other components, and (c) *scalability*, in order to capture large-scale effects in a lightweight manner. Existing solutions either fail to capture the spatial patterns in network traffic or introduce significant computational overheads and are unable to scale past a few servers.

In this paper, we present ECHO, a scalable and accurate modeling scheme that captures the spatial and temporal behavior of network traffic in large-scale DC applications. ECHO is derived from *validated* analytical models that enable it to concisely represent the network patterns of workloads, while provably guaranteeing low and upper-bounded errors. As part of ECHO, we present two models; first we examine a simple, distribution fitting model that captures and generates per-server network traffic by recognizing known distributions in network activity fluctuation. To capture the burstiness and *spatial patterns* of network load, e.g., server-to-server traffic, we propose a Markov Chain model that is topology-independent and locality of communication-aware and captures individual server interactions. Additionally, we make ECHO hierarchical, adjusting the level of detail in the model to the requirements of each application. Starting from groups of racks and increasing the level of detail down to individual servers, ECHO captures and recreates the spatial patterns of network activity in DCs with tens of thousands of servers. We perform a detailed validation study and show that the deviations between original and generated traffic are marginal for the network activity of two large-scale systems. We also verify that ECHO captures all the critical features of DC applications, such as spikes in network activity and inter, intra-rack communication. Additionally, we perform a detailed characterization of the temporal and spatial patterns of the network activity of DC applications in three DC deployments over a period of five months.

The rest of this paper is structured as follows. Section II discusses related work. Section III presents a description of the simple temporal model while Section IV includes a detailed characterization of the network activity of the DC applications. Section V presents the hierarchical Markov model and its validation against real network applications. Finally, Section VI presents topics for future work and concludes the paper.

II. Related Work

Network Workload Characterization: The network is one of the most widely characterized aspects of a workload since it reflects user patterns that emerge in the application. This characterization becomes more interesting for DC workloads running on tens to hundreds of thousands of servers. Although there is extensive prior work on network characterization for traditional applications [2], [7], [10]; in this paper we focus on related work in the context of large-scale DC systems.

Feitelson [12] presents a detailed characterization of network requests based on their stationarity, self-similarity, burstiness, and heavy tails; features that dominate DC workloads. Yu et al. [20] use a detailed profiling of the TCP/IP stack to troubleshoot network performance problems of multi-tier DC applications. They design a generic, app-independent profiler that monitors TPC at the socket-level and identifies performance bottlenecks within the same and across different connections. Ersoz et al. [11] also characterize the network traffic of a multi-tier DC. They observe that inter-arrival times and message sizes follow log-normal distributions, while service times fall within the Pareto distribution and show heavy tails at heavy loads. Benson et al. [5] analyze the features of network traffic in several cloud DCs classified per application type, in terms of temporal patterns, network and link utilization, congestion and packet drops and in [6] propose a fine-grain scheme for traffic engineering in these systems.

Atikoglu et al. [1] perform a workload analysis of Facebook's key-value store, *memcached*. They observe that GET requests by far dominate over SETs, while spatial locality widely varies across memcached servers. They also record user patterns that experience diurnal behavior and propose statistical modeling to extract the distribution of request inter-arrival rate. Also in the area of large-scale workload analysis, Shafiq et al [18] study the machine-to-machine (M2M) traffic in cellular networks which has similarities with the traffic of certain user-interactive DC applications. They characterize the temporal dynamics (e.g., diurnal behavior, burstiness), hot spots, application usage and data upload/download of a dataset from a large network service provider. Many of their findings are consistent with the behavior we present in Section IV for latency-critical DC applications. Using network characterization in a different scope, Gill et al. [13] measure DC network load to evaluate the system's reliability and characterize the components most prone to fail.

Network Workload Modeling: Similar to characterization, network modeling has attracted significant interest due to the user patterns of large-scale DC applications. Feitelson [12] apart from characterizing network loads, presents an overview of methods to model network request distributions. He suggests distribution fitting through the Kolmogorov-Smirnov test, to identify known distributions in network traffic fluctuation. Furthermore, he presents preliminary considerations on the correlation between task size, arrival rate and execution time when combining network and CPU modeling. Although his work presents strong arguments for the value of modeling temporal variations of network traffic, it does not present any validation of the proposed techniques against actual applications. In Section III we evaluate a similar distribution fitting-based model, but additionally validate it against real DC workloads. Building from this paper, Li [15] characterizes network and CPU-intensive applications running on large-scale grids. He analyzes features like job arrival rate, size, pseudoperiodicity and correlation of these features with execution time. He, then, proposes a two-phase approach to model these workload attributes. The first step consists of *Model-Based Clustering* which performs distribution fitting. The second step generates autocorrelations to create synthetic workloads that resemble the original load. Although this work provides some insight on the performance impact of request distributions in grids, it is computationally intensive and does not scale beyond a few nodes, making it inapplicable in large-scale DCs.

Barford et al. [2] study the characteristics of Web servers and propose a workload generator that recreates patterns with temporal fluctuation and request sizes similar to the original application. Joo et al. [14] propose network traffic modeling to identify and resolve performance bottlenecks in a small machine cluster. They compare two different models, an *infinite source-based* model that is user-invariant, i.e., all users send the same amount of data, and a *SURGE-based* model, where traffic varies per user. They observe that ignoring user variation and information about network topology causes significant inaccuracies in the generated workloads. Sengupta et al. [17] characterize the request arrival rates of a series of OLTP workloads and propose an analytical distribution fitting and self-similarity recognition model. They conclude that accurate modeling of network traffic can facilitate decision-making for a series of performance/energy-related optimizations, although their study is only limited in transaction-based applications running on a small cluster. Similarly, Danzig et al. [8] propose an empirical workload model based on statistical analysis of wide-area TCP/IP traffic. They use both request inter-arrival times to model traffic that follows known distributions and amount of data transfered to model burstiness. In the same spirit, Meisner et al. [16] model the network request arrival rates of a large-scale latency-critical online application with a G/G/1 queueing model and use it to generate representative loads, as inputs to a DC simulator. Finally, Tang et al. [19] propose a framework that captures non-stationarity, burstiness and request duration to model long-time network behavior. Based on this model, they develop MediSyn, a publicly available streaming media workload generator.

Overall, the common base in previous work is a focus on capturing temporal variations in network load using analytical models such as distribution fitting. Although this can provide

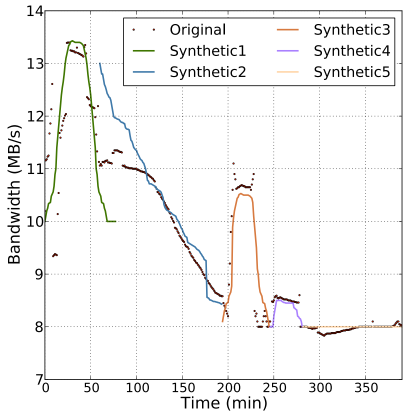

Fig. 1: Distribution fitting model validation.

insights on load variations and identify user patterns over time, it includes no notion of spatial patterns, which are a crucial part of DC applications. Hot spots due to load imbalance, inter-application traffic and server-to server communications are all features that impact critical design desicions in large-scale DCs and necessary for a modeling scheme to capture.

III. Single Server Temporal Model

We first focus on the requirements for simplicity and conciseness of the scheme, with a simple model that captures per-server network activity over time. We show which features of network load this model captures and why on its own it is not sufficient to model the behavior of DC applications.

We adopt a distribution fitting model which has been previously shown to be useful to capture the behavior of conventional network workloads [15], [17]. In this work we are validating its accuracy in the context of a real large-scale DC application. The model takes as input a network bandwidth trace from a single server and identifies known distributions (e.g., Gaussian, Poisson, Zipf, etc.) in the activity pattern. The output is a mathematical expression that is the superposition of identified distributions similar to Exp. 1.

$$BW_o = \sum_{i=0}^{N} (Distribution\ Expressions) =$$
$$\sum_{i=0}^{N_1} f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_i}{\sigma_i})^2}|_{t_{i_{start}}}^{t_{i_{stop}}} +$$
$$\sum_{i=0}^{N_2} f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}|_{t_{i_{start}}}^{t_{i_{stop}}} + \sum_{i=0}^{N_3} f_{other}|_{t_{i_{start}}}^{t_{i_{stop}}} + ... \quad (1)$$

where N is the number of identified distributions and N_i the number of individual distributions of each type.

We validate the accuracy of the model by comparing the network traffic generated based on the model against original traffic patterns. Fig. 1 shows this comparison for a Webmail workload running on a production-class server. The deviation between original and generated load is less than 4.9% on average, ensuring that the model accurately captures temporal variations in network load. In this case, the model identifies three Gaussian, one exponential and one constant distribution in the network activity. We have performed additional validation experiments with workloads that experience diverse activity patterns and verified the consistency of the results.

Although the distribution fitting model is a simple and comprehensive way to capture the load of individual servers, it is agnostic of the source and destination of traffic, therefore it cannot represent spatial effects, such as server-to-server communication. Additionally DC workloads often experience short bursty periods in their network activity [12], which cannot be accurately approximated by known distributions. The next Section describes the most critical features of network workloads in the large-scale that the model should accurately capture. To recreate them we adopt a different approach. The hierarchical Markov chain model described in Section V is a scalable, yet accurate solution that captures both temporal and spatial effects without becoming intractable in complexity.

IV. Temporal and Spatial Network Traffic Characterization

Previous work has established some trends for DC workloads, such as diurnal behavior and variation in activity between different time intervals, e.g., weekdays over weekends [1], [5], [11]. Here we validate these findings and additionally perform a detailed study on the spatial locality of network traffic in DC applications, as well as its fluctuation over time. We examine the network load of three production DC deployments; a system with several tens of thousands of servers from Microsoft running Websearch and two smaller systems with several hundred and a few thousand servers respectively running a single application. Specifically, the first system runs a combiner for query results, part of Websearch (Combine), while the second system extracts a snippet of information from backend Websearch servers and displays it to the user, in the top of the search results (Render).

Fluctuation of network activity over time: We show network traffic over time in the form of heatmaps. Each tick on the x-axis represents an interval of 5 minutes and each point on the y-axis, a server ordered by server id. Consecutive servers belong to the same rack in groups of 48 servers per rack. The colorbar on the right shows the range of network traffic bandwidth observed. Darker color in the heatmap represents higher network traffic. There are a few servers in each heatmap that exceed this range, however they represent a very small percentage of the total deployment, typically 0.1%-0.5%. Fig. 2(a) shows the fluctuation of activity per server in the large DC running Websearch over the period of one month (December 2011). As shown in the graph, the network load varies across groups of servers, with specific nodes experiencing high traffic, although mostly the system remains well load-balanced. Overall, the network traffic is very low; very few nodes exceed 0.0064MB/s, which is in agreement with the extensive overprovisioning present in large-scale deployments of latency-critical applications such as Websearch [3], [4]. Additionally, the traffic over time remains mostly self-similar, consistent with well-known DC application characteristics [1].

The diurnal patterns in network activity, i.e., difference in load between day and night, become more clear in Fig. 3(a) and (c) which show per-server traffic for Combine and Render. Every dark vertical band representing the hours of the day is followed by a lighter interval (e.g., in the x-axis: vertical band at range 970-1030) representing the low activity period of the night. There are 31 dark and 31 light bands in total in the graph representing each 24h period in the month. Compared to Websearch, both Combine and Render have significantly higher network traffic, as a result of the lookups necessary to extract and aggregate search results. Between Combine and Render, the former has slightly higher network activity, since larger chunks of data are transfered between servers. To better visualize the diurnal patterns in network activity and their change, in Fig. 4 we plot network load over time, averaged across all servers. Fig. 4(a) shows the fluctuation in load for Websearch across five consecutive months (November 2011-March 2012). Overall, the patterns are similar, although their magnitude varies across different months, with significantly higher loads in December and January. Fig.4(b) shows the per-week breakdown of network load for December. Again, diurnal patterns are present, with load being 10-15% higher in the last two weeks of the month, and 15-18% higher in the days of the weekend, compared to weekdays. Finally, Fig. 4(c) shows the per-day breakdown for the last week of the month. The load experiences two peaks, one from 12pm to 6pm and another from 8am to 12pm while progressively decreasing in the hours of the night.

Although the time aspect of DC applications has been extensively studied, the same is not the case for the locality of communication in network applications. This study has three types of contributions; first, it can verify that the application takes advantage of *data locality* by confining most requests to servers of the same or neighboring racks to reduce latencies, second, in the presence of multiple applications, it can provide insight to the *scheduler*, on how to assign machines to jobs to confine inter-application traffic to neighboring nodes and third, it can verify that *load balancing* prohibits the formation of extensive hot spots. In this work, we examine both the average locality of network traffic and its fluctuation over time to provide insights on these issues.

Spatial locality of network activity: Fig. 2(b), Fig. 3(b) and Fig. 3(d) show the server-to-server network traffic for Websearch, Combine and Render. Both the x and y axes represent servers and the graphs are symmetric over the minor diagonal since, for example, traffic between servers 2 and 3 is the same as traffic between servers 3 and 2. This includes both sent and received traffic for each pair of servers. More prominently in Websearch, but for the other applications as well, the majority of traffic is confined within servers of the same or consecutive racks, seen by the darker colors along the diagonal. For all three applications, especially Combine and Render, there is a small number of servers that talk to most of the machines in the system. These are likely servers running the cluster scheduler, aggregators (in multi-tier Websearch) or monitoring systems.

In the smaller clusters of Combine and Render, we observe less network locality, as several servers, especially in the first few racks, exchange requests with machines outside their rack. This is consistent with the functionality of the two applications, aggregating and caching or displaying results extracted from search queries. Data used to service these queries may require a large number of shards, spanning multiple servers. Overall, we observe that in Websearch, which is dominated by query latency, the job scheduler is data locality-aware and tries to minimize long request round trips to servers that are far from each other.

Finally, we examine the actual network traffic map for Websearch in Fig. 2(c), where each shaded region corresponds to a rack, with 48 servers within each region. All servers in a rack are connected to the same switch, while servers in different racks are connected to different switches. As seen in the figure, with the exception of few, most racks have similar levels of network load, verifying that the load balancer assigns work to machines preserving fairness, while the small number of racks which have slightly higher network traffic are the ones that talk to most of the servers in the DC.

Fluctuations in the locality of network activity: Apart from examining the average locality of network activity, we look at how these patterns change over time. Fig. 5 shows the server-to-server network activity for Websearch over a period of five consecutive months in 2011 and 2012. In most cases the patterns remain consistent across different months with a slight surge in traffic in December and January, which is in agreement with our findings for the activity fluctuation over time for the same period (Fig. 4). When comparing this to the fluctuation in locality observed in Combine in the same period (Fig. 6) there are significant differences. Unlike Websearch, Combine experiences significant differences in its locality over different months, probably the result of changes in the application's structure and functionality. For example, while in November and December there is a significant number of servers polling a large fraction of the system, in the following months the requests become progressively more localized in servers of the same or neighboring racks. March also experiences an interesting access pattern with high localized network activity in specific servers (range 180-250).

Although at the month granularity Websearch seems invariant in its locality patterns, when moving to a finer granularity we see that it also experiences fluctuations in the locality of its network activity. Fig. 7 and 8 show the server-to-server traffic for two weeks of December and three days of the second week respectively. At the granularity of individual days, we observe that there are significant differences in the locality of accesses, with different subsets of the DC being more active than others; however, these fluctuations get hashed out at large time intervals.

Overall, we observe that the network activity of large-scale applications changes both in time and space, and for a model to provide useful insight in the behavior of the application, it should capture the workload accurately across both these axes.

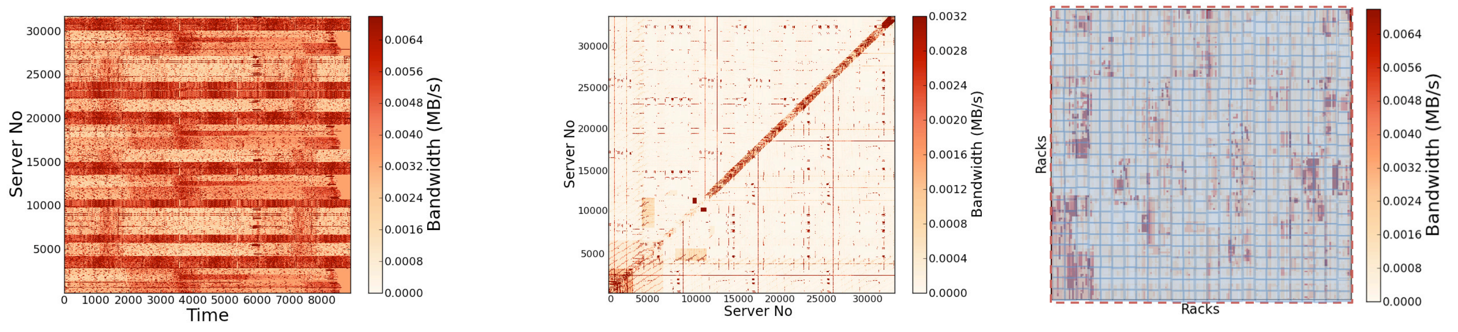
Fig. 2: (a) Per-server network traffic over a 1 month period, (b) server-to-server traffic (locality of network activity) and (c) rack-level network traffic map across the entire DC for Websearch.

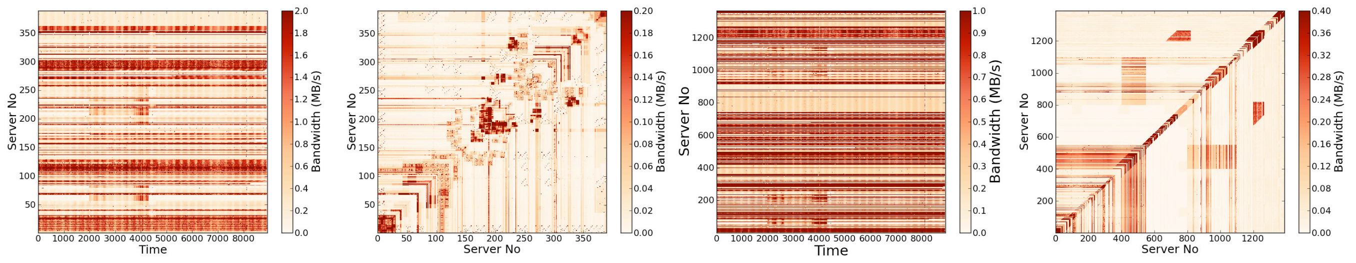
Fig. 3: Network traffic over time and server-to-server traffic for Combine and Render.

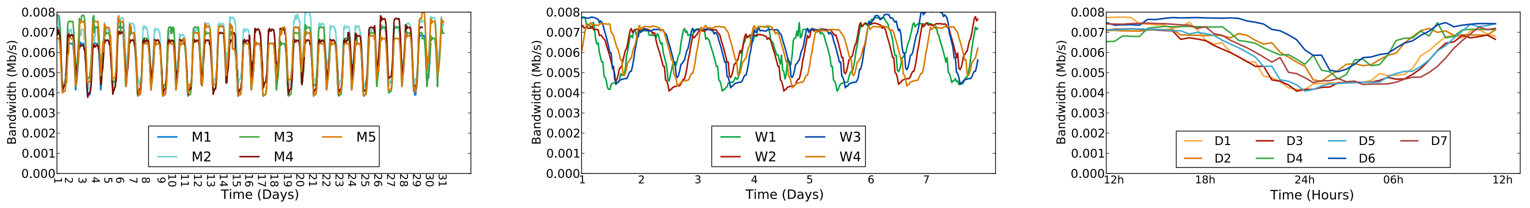

Fig. 4: Network activity averaged over all the servers in the system across (a) five months, (b) four weeks and (c) seven days.

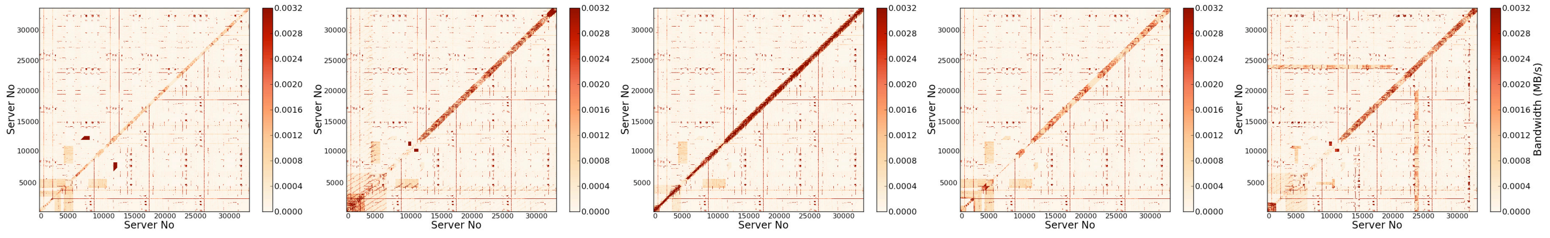
Fig. 5: Server-to-server traffic for Websearch over a period of five consecutive months in 2011 and 2012.

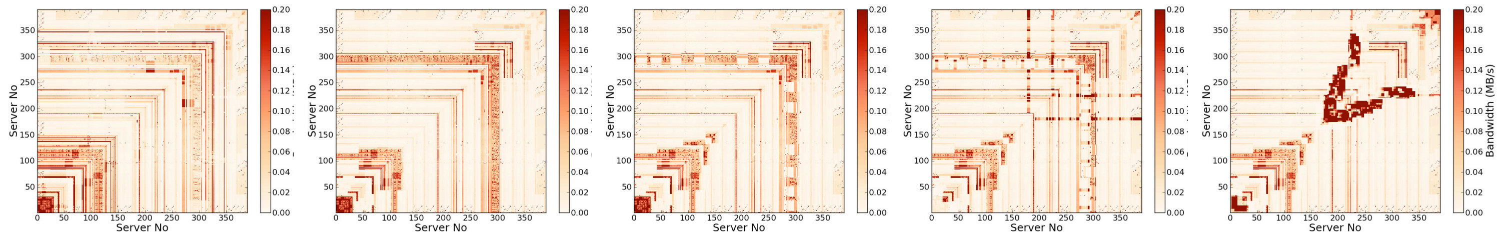
Fig. 6: Server-to-server traffic for Combine over a period of five consecutive months in 2011 and 2012.

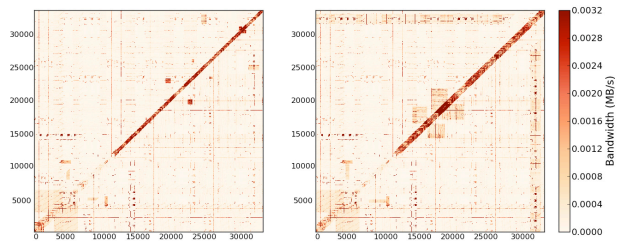

Fig. 7: Spatial locality of network activity across two different weeks in December 2011.

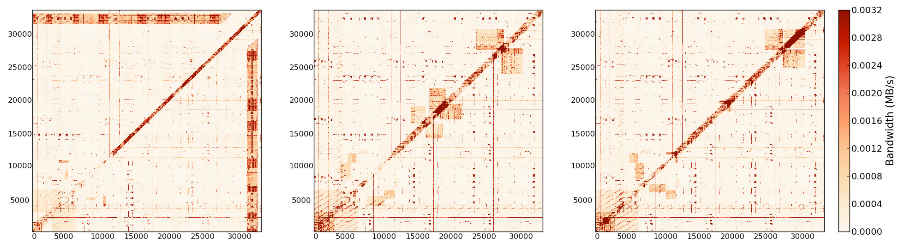

Fig. 8: Spatial locality of network activity across three different days in December 2011.

V. System-Wide Spatial Model

A. Overview

From the previous Section there are several network activity features that are critical for a workload model to accurately capture; (a) the *average activity* (per-server and system-wide), (b) the *fluctuation of activity over time*, both bursty and with defined intervals, (c) the *locality of activity*, i.e., spatial patterns across the system and (d) the *interactions* between specific servers or racks. This is a wide and often conflicting set of requirements to be met; for example a model that captures spatial distribution of load might not capture temporal variations accurately.

B. Design

To address these requirements we propose ECHO, a scalable modeling scheme that captures both temporal and spatial network patterns. ECHO leverages robust analytical models that enable it to provide strict error bounds on the accuracy of the representation. Specifically, it is driven by a set of Markov chains that are trained on real input traces from production DCs and provide a probabilistc framework to both characterize and recreate network activity patterns. The selection of a Markov model is based on its ability to concisely and accurately capture the behavior of a wide spectrum of systems, the identification of more dominating system states (states with higher probabilities) and the information on transitions between states. Also, Markov models have previously been shown to accurately capture the behavior of DC workloads in other subsystems, such as storage activity [9].

To tackle the scale of modern DCs, ECHO is hierarchical, starting from groups of racks, and modeling network activity down to racks and individual servers. This way the level of granularity is adjusted to the specific patterns of a given application. ECHO is topology-independent, since the aggregation of state in the model is not tied to a specific network organization, therefore it is portable across different topologies, provided no radical changes in the system's organization. Also, the model is workload-independent and although the exact structure and probabilities vary across different workloads, the model's principal design is preserved. Fig. 9 shows the hierarchical probabilistic model for a system with four groups of racks. Each state represents different entities in each level. In the highest abstraction, a state is a group of racks, while at the second level it represents a rack, and at the lowest level it represents an individual server. The number of groups of racks or racks in each group is a configurable parameter of the model, while the number of servers per rack is a design parameter of the system. A transition between any two states represents the probability/portion of network traffic between two entities that has certain characteristics. These characteristics correspond to the size of network requests, the type and the inter-arrival times that separate them, or their burstiness. The Markov chain defines such probabilities as:

$$p_{ij} = Pr[X_j|X_i] =$$

$$Pr[Server_j \leftarrow Server_i \ \mid MB/s,\ rd/wr,\ int.\,time] \quad (2)$$

where there is a p_{ij} probability that $Server_i$ interacts with $Server_j$ over the network with specific bandwidth (MB/s), request type (rd/wr) and inter-arrival time. Although the use of both inter-arrival times and probabilities might seem redundant, it enables modeling both requests that follow well-defined distributions (in which case the two metrics are almost equivalent) and bursty behavior (when the inter-arrival rate is very high, but the probability of the traffic occuring low).

For the clarity of the representation some transitions are omitted from Fig. 9. Based on the workload characterization of Section IV, we observe that the majority of network traffic, with a few exceptions, is confined within the servers' corresponding rack. This motivates the use of a hierarchical rather than a flat model where all transitions are explored. As seen in Fig. 9 transitions between large states remain coarse-grain when moving to lower levels of granularity, while transitions within large states are expanded. This improves the scalability of the scheme, by enforcing upper bounds for the transition count and prevents a complexity explosion when scaling up to thousands of servers, while conveying the same amount of information about the workload. Comparing a flat and hierarchical model for a system with 100 racks reveals a 90% reduction in transition count for the hierarchical model. Additionally, because of the design of the model, the representation of network activity becomes very concise; while raw traces require GB and often TB of storage, the hierarchical model of the 100-rack system only takes a few KB of storage. Even the use of compression techniques to reduce the size of trace data cannot easily offer a 6-9 order of magnitude space

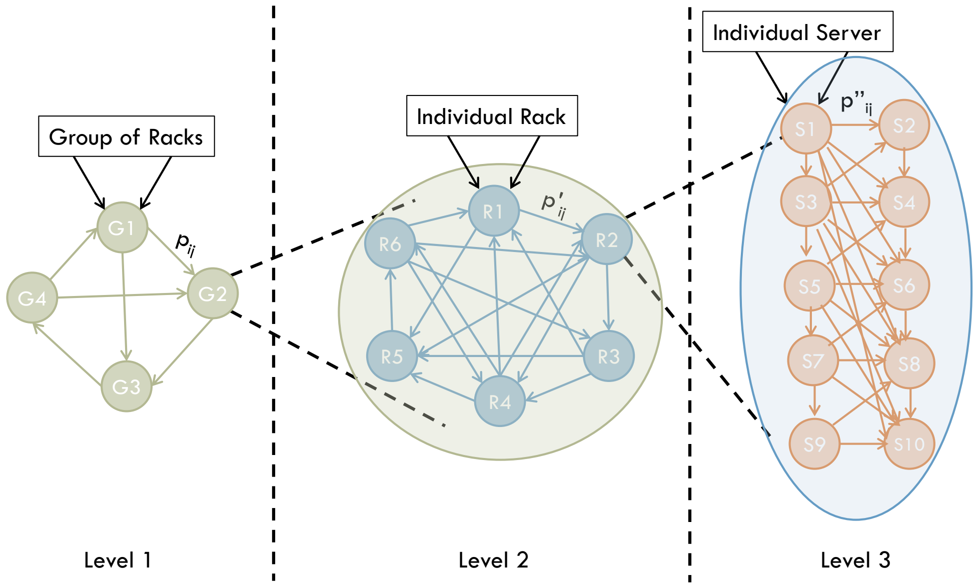

Fig. 9: Schematic of the hierarchical spatial Markov chain model.

reduction, while it would require additional analysis to explore activity patterns in the compressed data.

The model described so far captures both spatial and temporal patterns in a DC for a given time frame. However, network activity is highly volatile switching from low to high load frequently and experiencing short periods of high burstiness in the requests. To capture such evolving patterns we rely not on one, but on a set of Markov chains, each one corresponding to a different interval of the application's execution. The number of models is a configurable parameter, depends on the characteristics of the application and since individual models can be created in parallel, higher disaggregation in time not only does not increase, but decreases the overhead of the modeling process. For example, to create a single Markov chain a system-wide bandwidth trace is required. Parsing the entire trace, although only done once can introduce some overhead in the modeling process. However, partitioning this procedure to create multiple Markov chains for different time periods and processing them in parallel, both reduces the training overhead and improves the convergence between original and generated load. When generating a synthetic workload, the different models are activated in a Round-Robin fashion based on the time frame they correspond to, e.g., when there are separate models for each 12h period of a day, 50% of the time the generated workload complies with the first model and 50% of the time with the second model. Finally, although the model is non-deterministic, which means that subsequent runs may be different, we verify that each one of them maintains close resemblance to the original workload.

C. Validation

For our validation we focus on four main metrics, as discussed in Section IV; (i) average network traffic, i.e., *how much traffic is generated on average irrespective of temporal and spatial patterns*, (ii) per-server activity fluctuation over time, i.e., *how much traffic*, (iii) spatial patterns of network activity, i.e., *to whom is the traffic sent* and finally a more fine-grain metric, (iv) individual server interactions, i.e., *how much traffic, to whom and when*. We show validation results for the hierarchical spatial model using as input real network activity traces from two production DC deployments with hundreds to tens of thousands of servers. Unless otherwise specified all results are using the 3-level hierarchical model. Additionally, experiments shown in this work are repeated over five runs to verify the consistency of the results. Also, although DCs in this study host a single application, the model can be applied to space-multiplexed workloads as well, since typically requests are tagged with the application id that initiated them. In that case, a different model for each of the co-hosted applications would be created.

Average activity validation: We evaluate the model against two types of systems. First, we validate the model in a small test cluster of a single rack with 24 servers. The workload is the network traffic of servers hosting the webmail of an academic institution. We examine four two-hour instances of this load; Load A is the load on the morning of a weekday, Load B the load on the afternoon of a weekday, Load C the load on a night of a weekday and Load D the load on a weekend. Fig. 10 shows the comparison between original and synthetic traffic generated based on the spatial model for Load A (the results are similar for the other loads) and the table in Fig. 11 the statistical metrics that quantify this deviation. In all cases the difference between original and synthetic workload is less than 7% validating the accuracy of the model.

Second, we validate the model against real DC workloads on two production systems; the DC running Websearch on several tens of thousands of servers and the smaller cluster running Combine. The temporal patterns in Section IV reveal that load changes significantly both throughout a 24h period, and between different weekdays compared to the weekend. Based on these patterns we create a set of 10 models for

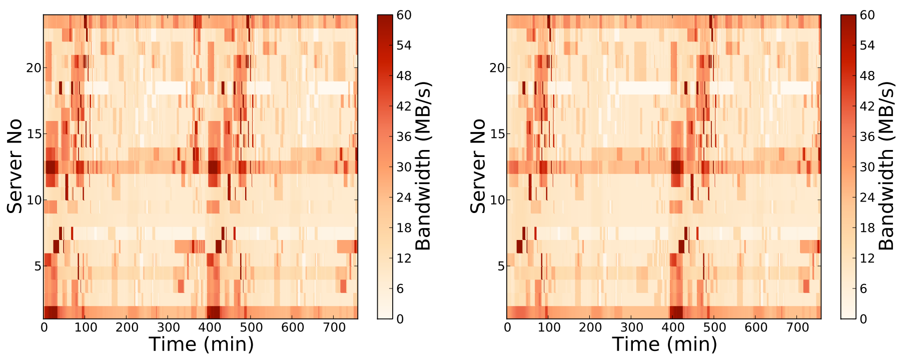

Fig. 10: Per-server bandwidth comparison between original and generated network traffic for a one-rack system running Webmail at load A.

Server Percentage	Average Deviation			
	Webmail			
	LoadA	LoadB	LoadC	LoadD
25%	3.8%	4.3%	2.3%	2.9%
50%	4.2%	5.6%	2.5%	3.4%
75%	4.5%	6.2%	2.7%	3.5%
95%	5.5%	6.3%	3.1%	4.2%
All Servers	5.8%	6.3%	3.0%	4.1%

Fig. 11: Validation of average statistics between original and generated network load.

Application	Average Deviation for Percentages of Servers				
	25%	50%	75%	95%	System-Wide
Websearch	7.2%	5.9%	4.2%	3.9%	3.8%
Combine	8.5%	6.9%	5.3%	4.4%	4.4%

TABLE I: Validation of average statistics between original and generated network load for Websearch and Combine.

each system; 4 models from six hours periods of two different weekdays $(2 \cdot 4)$ and 2 models for 12h periods of a day of the weekend. This choice is configured to the features of the specific applications and can be different for other workloads. We first validate the average statistics of network activity between original and generated workloads. Table I shows the deviation in average traffic both for fractions of servers and system-wide. The first 4 columns show the deviation for the 25, 50, 75 and 95 percentages of servers, sorted by increasing average network load and the last one shows the average deviation across the entire system. In all cases, the deviation is marginal, with the system-wide being lower than the ones for server subsets for both applications. We also observe that the lower the load of a server the higher the deviation between original and synthetic workload, while for higher loads the deviations are lower. Additionally the larger the modeled system the lower the deviation.

Activity fluctuation validation: We validate the accuracy of the model in capturing the fluctuation of network activity by generating synthetic workloads for subsets of the two production DCs, Websearch and Combine. There are two reasons for not validating the model against the entire DC; first, the unavailability of an equally large system to run the generated workload and more importantly the fact that although possible, system-wide generation of a workload obscures some of the value of the model, which is identifying interesting patterns or performance issues in network traffic and reproducing them independently, without full application deployment. The validation process works as follows:

- We select subsets of servers in the original workload that present interesting traffic patterns.
- We create models that are trained on network activity traces from each subsystem.
- We generate a synthetic workload for each subsystem based on the corresponding model.
- We compare the network traffic between original and synthetic workload.

Fig. 12(a) shows this validation for Websearch and Fig. 12(b) for Combine using as input the network activity over a one month period (December 2011). This activity for Websearch includes traffic from multiple components of the application, such as indexing, advertisements and answers to user queries, while for Combine it represents network traffic from a single application. We have selected four server subsets in each case, representing machines with both low and high activity, as well as static or varying traffic patterns over time. The figures visually demonstrate the similarity between original and synthetic patterns, while the corresponding Table II quantitatively confirms these findings. In all cases, the errors both for percentages of servers and across the entire sample are lower than 7% for Websearch and 8% for Combine, verifying the accuracy of the model in capturing time variations. In general, errors for activity fluctuation increase as we move to larger server percentages, in contrast to what we saw for average activity errors. This happens because mildly-fluctuating activity of lightly-loaded servers is easier to model, than server subsets with drastic load changes. Additionally, short time skews between original and modeled workload, create large errors for activity fluctuation, but are hashed out when validating average statistics.

An important point when modeling system subsets is that traffic is not necessarily self-contained in them. For example, if a subsystem includes servers 1-10, it is not necessarily true that server 1 does not interact with server 11 and vice versa. Therefore to maintain the accuracy of per-server load when choosing parts of the system, we need to ensure that the external traffic is also represented in the model. We do this by introducing a single ”cloud” node that generates and receives all traffic from and to servers in the subsystem. Fig. 15 shows how this external node is integrated in the model. Essentially the model has one additional state which is responsible for the network traffic outside the selected subsystem.

Application	Average Deviation (%) for Percentages of Servers in each Subset																			
	Subset S1					**Subset S2**					**Subset S3**					**Subset S4**				
	25	50	75	95	100	25	50	75	95	100	25	50	75	95	100	25	50	75	95	100
Websearch	1.2	2.4	5.7	6.8	6.6	3.9	3.8	4.7	5.3	5.2	3.2	3.6	3.7	4.1	4.0	7.2	6.7	6.8	7.1	7.0
Combine	2.3	3.5	3.6	4.4	4.2	4.2	3.8	3.6	3.7	3.8	5.1	4.8	4.9	4.6	4.5	7.6	7.9	8.1	8.0	8.0

TABLE II: Validation of generated network activity over time for server subsets in two DCs.

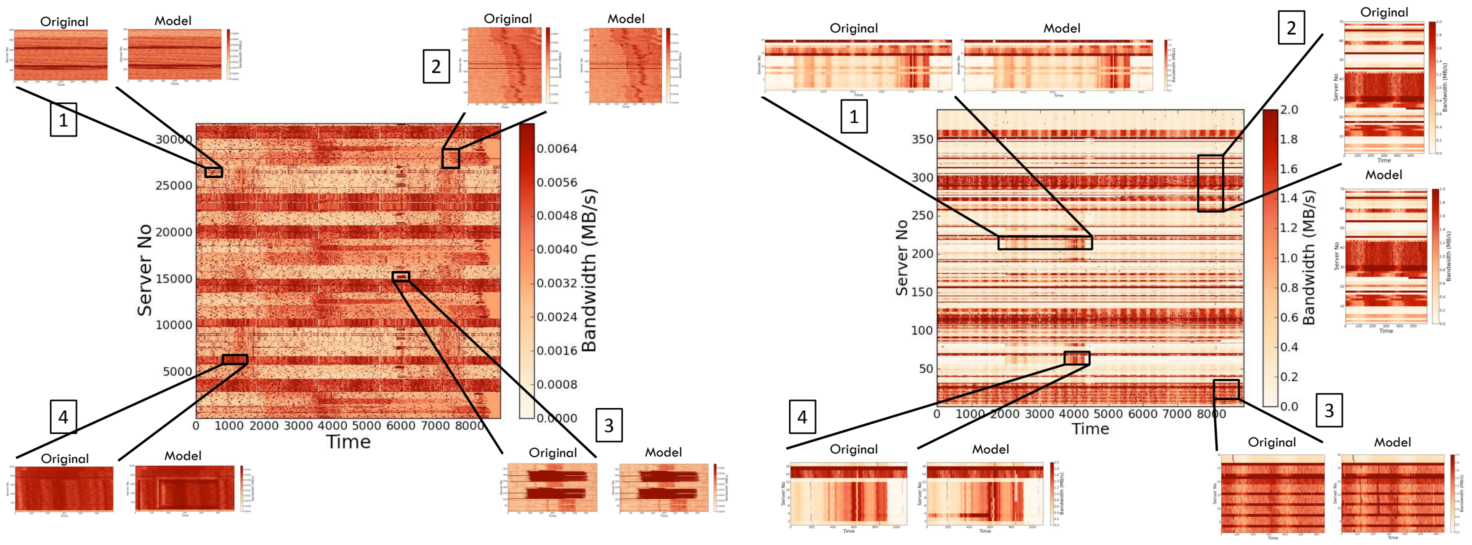

Fig. 12: Spatial model validation for server activity over time. Fig. 12a shows the validation of ECHO against subsets of the large-scale DC running Websearch and Fig. 12b a similar validation for Combine running on a smaller DC.

Application	Average Deviation (%) for Percentages of Servers in each Subset																			
	Subset S1					**Subset S2**					**Subset S3**					**Subset S4**				
	25	50	75	95	100	25	50	75	95	100	25	50	75	95	100	25	50	75	95	100
Websearch	9.5	5.2	3.4	3.2	2.7	6.2	5.4	4.9	4.6	4.6	4.2	4.1	4.3	4.3	4.4	10.8	9.9	7.6	8.1	8.2
Combine	6.5	4.9	4.3	2.4	2.4	3.4	3.5	4.7	4.6	4.5	5.8	6.7	7.3	6.3	6.4	11.2	10.8	8.2	5.6	5.4

TABLE III: Validation of generated network traffic map for server subsets in two DCs.

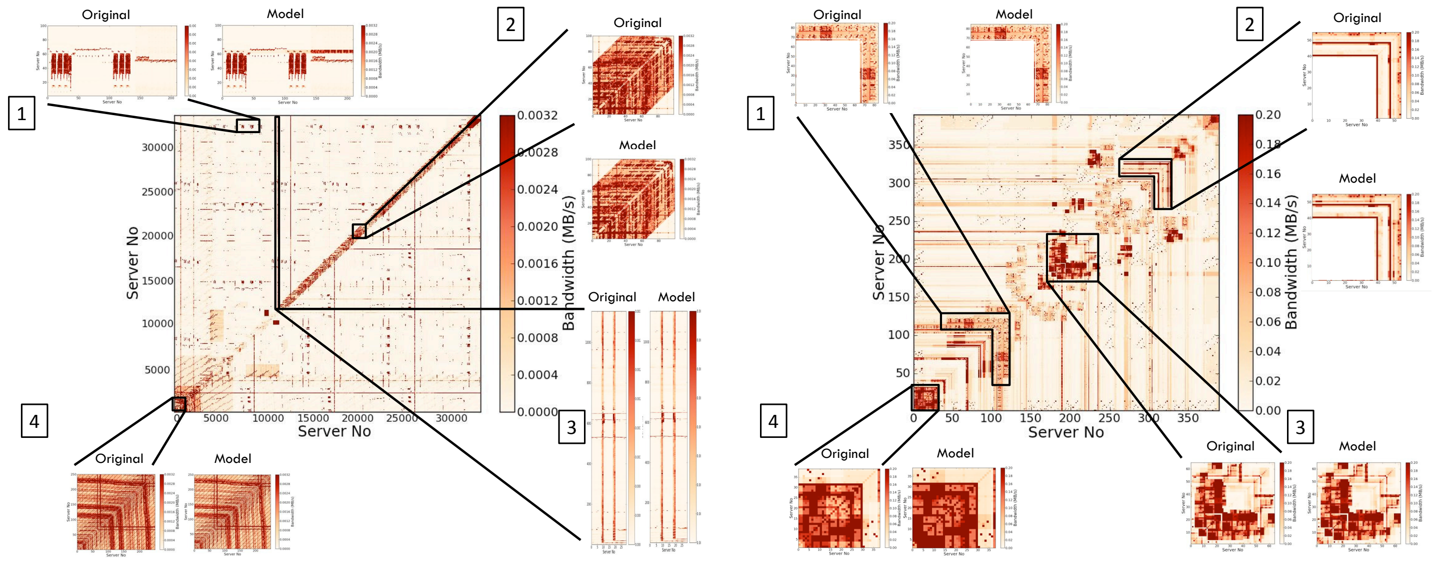

Fig. 13: Server-to-server network traffic map validation for server subsets of (a) Websearch and (b) Combine.

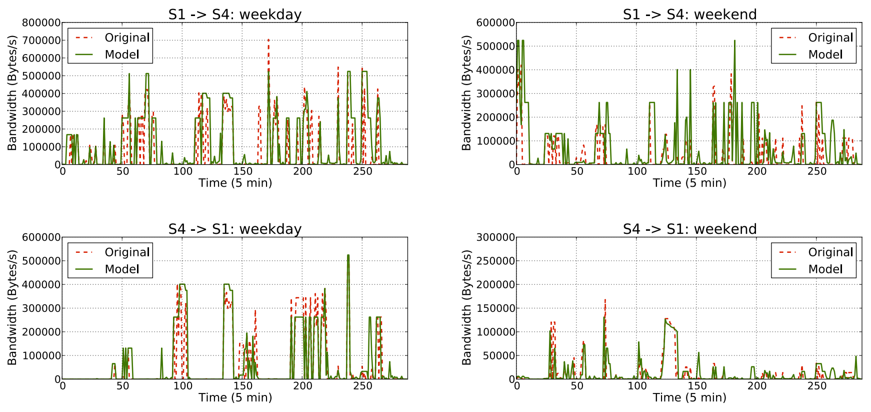

Fig. 14: Inter-server communication validation for a day of the week (Fig. 14a) and a day of the weekend (Fig. 14b) for two Websearch servers.

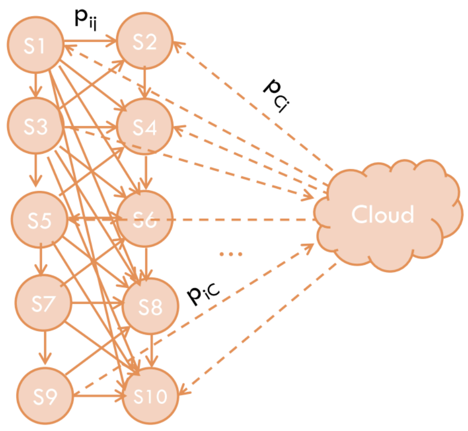

Fig. 15: Modified model for server subsets. The "Cloud" node represents traffic generated from or directed to external nodes in the system.

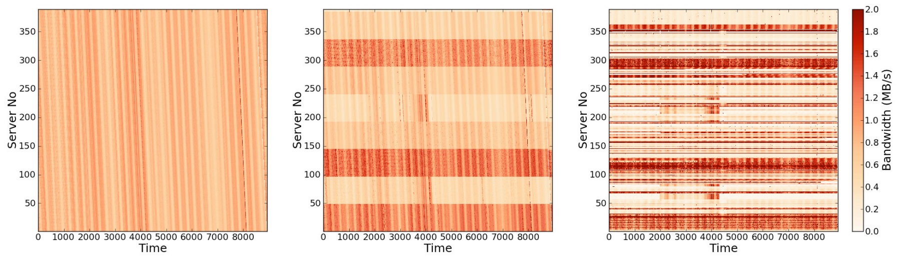

Fig. 16: Model representation when moving from single-level to two and three-level hierarchy for a DC running Combine.

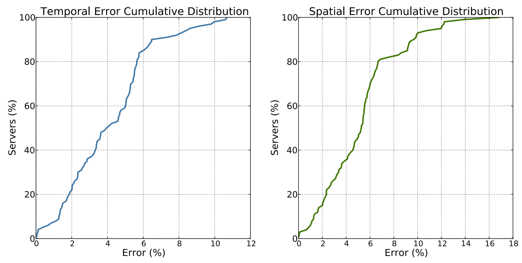

Fig. 17: CDFs of error distribution across servers for (a) network activity fluctuation and (b) network activity locality.

Validation of locality of network activity: A critical goal of the model's design is capturing spatial patterns in network activity. We verify this with a similar experiment to the one previously described. We select four subsets of servers from the server-to-server traffic maps for Websearch and Combine and generate synthetic workloads based on the corresponding models. Fig. 13 shows a visualization of the similarities between original and synthetic subsets and Table III presents the quantitative metrics that confirm these similarities. For both systems the deviations are low, with higher load servers typically experiencing smaller inaccuracies. Overall the deviation is less than 10% for Websearch and 11% for Combine.

Finally, Fig. 17 shows the error CDFs for the temporal (Fig. 17a) and spatial (Fig. 17b) patterns of the generated workload against the original application. The y-axes in both graphs represent server percentiles. The errors reported are for the large-scale DC running Websearch. From the figure, the errors for spatial patterns are slightly higher than for temporal patterns, but it all cases the 50^{th} percentiles of servers have less than 4% error in temporal and less than 5.2% in spatial patterns while the 90^{th} percentiles of servers have less than 7.3% in temporal and 9.8% in spatial patterns. Only few outliers experience errors larger than 10%, validating the accuracy of the modeling scheme.

Validation of server interaction: Finally, we validate the accuracy of the model in capturing network interactions between specific servers. Potentially this is the highest source of inaccuracy for the model given its probabilistic nature. Fig. 14 shows the comparison between original and synthetic workload for the isolated network activity of a server pair across two different days (a weekday and a weekend day). The upper figures show traffic from Server A to Server B and the lower figures traffic from Server B to Server A. The two servers are chosen randomly across the pool of servers of the large Websearch cluster. As the graph shows, the hierarchical design and temporal variance of ECHO enables close resemblance for the point to point network traffic. The deviation between original and synthetic workloads is 12.2% for the weekday, on average and 8.9% for the weekend, which is higher than the system-wide deviation but still a very close approximation of the original traffic. Most of the error occurs when very low traffic in the original application is not reflected in the generated workload. We have verified the consistency of these results with different server pairs.

Overall, the network load captured and generated by the model deviates marginally from the original application in all four critical characteristics, with average deviation being smaller than the other metrics and the recreation of temporal patterns being slightly more accurate than that of spatial patterns.

Sensitivity to model granularity: Finally, we examine the impact of hierarchy on the accuracy of the model. For the small DC running Combine we create three different sets of models; the first has information only at the granularity of groups of racks (level 1), the second at the granularity of individual racks (level 2) and the last one captures the workload at individual server granularity (level 3). Fig. 16 shows the network traffic generated by the model as more levels are added in the representation. Adding more levels significantly improves the accuracy of the model, as reflected in the corresponding deviations which are: 28% for the level-1, 9.1% for the level-2 and 4.4% for the level-3 model.

VI. Conclusions

We presented ECHO, a concise and scalable modeling and generation scheme for network traffic in large-scale DCs. ECHO captures both the temporal and spatial characteristics of large-scale applications and recreates per-server and system-wide network traffic maps. ECHO is driven by robust analytical models which allow us to provide strong guarantees on the accuracy of the workload information. We have validated the accuracy of these models against real DC applications and have shown marginal deviations between original and generated traffic. Thus, ECHO can be used to derive confident decisions on DC-related system studies. Specifically, we have studied the temporal and spatial patterns of two large-scale production DCs running Websearch. As part of future work, we plan to extend the applicability of ECHO in areas such as workload migration and server consolidation.

Acknowledgements

We sincerely thank Daniel Sanchez, the Microsoft Cosmos team and the anonymous reviewers for their useful feedback on earlier versions of this manuscript. Christina Delimitrou was supported by a Stanford Graduate Fellowship.

References

[1] B.Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, M. Paleczny. "Workload Analysis of a Large-Scale Key-Value Store". *In SIGMETRICS, 2012.*

[2] P. Barford and M. Crovella. "Generating Representative Web Workloads for Network and Server Performance Evaluation". *In SIGMETRICS, 1998.*

[3] L. Barroso. "Warehouse-Scale Computing: Entering the Teenage Decade". *ISCA Keynote, SJ, June 2011.*

[4] L. A. Barroso, U. Holzle. "The Datacenter as a Computer". *Synthesis Series on Computer Architecture, May 2009.*

[5] T. Benson, A. Akella, D. Maltz. "Network Traffic Characteristics of Data Centers in the Wild." *In Proc. of IMC 2010, Australia, 2010.*

[6] T. Benson, A. Anand, A. Akella, M. Zhang. "MicroTE: The Case for Fine-Grained Traffic Engineering in Data Centers". *In CoNEXT, 2011.*

[7] M. Calzarossa and G. Serazzi. "Workload characterization: A survey". *In Proc. of the IEEE, 81(8):1136-1150, 1993.*

[8] P. Danzig, S. Jamin et al. "An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations". *Wiley, Internet Journal, 1992.*

[9] C. Delimitrou, S. Sankar, K. Vaid, C. Kozyrakis. "Decoupling Datacenter Studies from Access to Large-Scale Applications: A Modeling Approach for Storage Workloads". *In Proc. of IISWC, TX, 2011.*

[10] A. B. Downey and D. G. Feitelson. "The Elusive Goal of Workload Characterization". *In Proc. of SIGMETRICS, 1999.*

[11] D. Ersoz, M. Yousif, C. Das. "Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center". *In Proc. of ICDCS 2007.*

[12] D. Feitelson. "Metric and Workload Effects on Computer Systems Evaluation". *In Computer, 36(9):18-25, 2003.*

[13] P. Gill, N. Jain et al. "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications". *In Proc. of SIGCOMM, 2011.*

[14] Y. Joo, V. Ribeiro et al. "TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations". *In Proc. of SIGCOMM, 2001.*

[15] H. Li . "Realistic Workload Modeling and Its Performance Impacts in Large-Scale eScience Grids". *In IEEE TPDS, vol. 21, no. 4, 2010.*

[16] D. Meisner, J. Wu, T. F. Wenisch. "BigHouse: A Simulation Infrastructure for Data Center Systems". *In Proc. of ISPASS, 2012.*

[17] S. Sengupta and R. Ganesan. "Workload Modeling for Web-based Systems". *In Proc. of CMG, 2003.*

[18] M. Shafiq, L. Ji et al. "A First Look at Cellular M2M Traffic - Large Scale Measurement and Characterization". *In SIGMETRICS, 2012.*

[19] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat. "MediSyn: A Synthetic Streaming Media Service Workload Generator". *In NOSSDAV, 2003.*

[20] M. Yu, A. Greenberg et al. "Profiling Network Performance for Multi-Tier Data Center Applications". *In Proc. of NSDI, 2011.*