Challenges and Opportunities for Managing Data Systems Using Statistical Models

Yanpei Chen, Archana Ganapathi*, Randy Katz
University of California, Berkeley, *Splunk
{ychen2,randy}@eecs.berkeley.edu, *aganapathi@splunk.com

Abstract

Modern data systems comprise of heterogeneous and distributed components, making them difficult to manage piece-wise, let alone as a whole. Furthermore, the scale, complexity, and growth rate of these systems render any heuristic and rule-based system management approaches insufficient. In response to these challenges, statistics-based techniques for building gray or black box models of system performance can better guide system management decisions. Although statistics-based approaches have been successfully deployed, a single model is often inadequate to capture intricacies of a single workload on a single system. The problem is exacerbated with multiple heterogeneous workloads super-positioned on a consolidated system. An even greater challenge is to translate the behavioral correlations found by statistics into insights and guidance for designing and managing even more complex data systems. In this article, we reflect on recent work on using statistics for data system modeling and management, and highlight areas awaiting further research.

1 Introduction

Large scale data systems are becoming ever more important. This is driven by increasingly economical access to large scale storage and computation infrastructure [4, 3], ubiquitous ability to generate, collect, and archive data about the physical world [11], and growing statistical literacy across many industries to consume, understand, and derive value from large datasets [1, 7, 15, 14]. The increasing ability to generate, record, and understand large scale data about the physical world is symbionic with the rising expertise to do the same for large scale data systems. Innovations in one area pioneer and inspire mirror innovations in the other. Statistical models have emerged as an essential tool for extracting knowledge about both physical and technology systems. This continues the historical relevance of statistics to scientific knowledge creation, in terms of inference, experiment design, hypothesis testing, and other ways to test and extend the limits of knowledge.

This technology context have led to the confluence of statistics and system design. In the past decade, a body of work emerged from both researchers and practitioners to use statistical techniques to mine data, model systems, understand systems, and manage systems [18, 9, 6, 13, 8, 12]. This effort is in response to the increasing complexity of data systems and the workloads they service, both composed of heterogeneous and often distributed components. Domain specific heuristics no longer scale, and no single domain expert can be

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

consulted for all system-related decisions. An effort to scientifically understand these systems brings rewards far outweighing the costs of re-engineering and the penalties of mis-engineering. Using statistical models becomes essential for designing and operating these systems.

There are a variety of commonly used statistical techniques, ranging from simple linear regression to more complex high-dimensional analysis. These techniques vary greatly in the amount of system specific knowledge they require and the system visibility they provide. A key challenge arises in identifying the boundaries of the system being modeled, what specific technique to use, how to adapt the statistical technique to serve the specific system management problem, how to evaluate the statistics-driven optimizations, and how to translate statistics to new knowledge about the data systems.

This article discusses these challenges and how they have been partially addressed by recent research. We focus on a few case studies to highlight in depth the challenges associated with defining system boundaries (Section 2), building statistical models (Section 3), evaluating statistical optimizations (Section 4), and turning statistics to new knowledge (Section 5). The case studies are mostly drawn from the authors' own work, because we are familiar with their limits. We highlight open problems and identify research opportunities that would benefit the general data system management community.

2 Defining a System

A *system* is a conceptual unit to which we can feed input, modify configurations, and observe performance under the given input and configuration. System input can be a single request, or a set of requests, often referred to as *workload*. Various metrics can characterize individual requests or entire workloads, ranging from request type, request size, to interarrival rate between requests, and others. The scale and mix of such requests often control how much work a system must perform, what kind of work they perform, and which configurations are the most appropriate. For example, in a relational database, queries serve as the input and sets of queries form a workload. The number, type, and arrival pattern of queries dictate the work required of the database, which can then be tuned for a particular workload.

System *configuration* often exposes tradeoffs to optimize system performance. For example, configuring a machine to use a subset of its cores for a particular application versus all its cores for that application would allow users to trade response time for resource sharing across different workloads. For another example, data systems often have logical configuration parameters to control data layout. One can control the layout of data among the nodes on a parallel relational database system, and different layout policies have an impact on resource utilization and query execution time.

System *performance* is an umbrella term that can be used for any observed behavior that is a consequence of running an input workload under a particular configuration. Some examples of performance metrics include execution time, CPU utilization, cache hits/misses, disk I/Os and latency. These metrics may be collected periodically, every minute for example, aggregated over a time window, such as average CPU utilization over a one minute window, or aggregated for an entire workload, as in the case of a performance summary for a database.

This simplistic definition of a system is extensible. A system composed of multiple subsystems has a set of input and configuration parameters for the entire system, which divides into input and configuration parameters for each subsystem. The performance of the entire system is a composition of the performance of each subsystem. Although the definition is simplistic, modeling such complex multi-component systems is non-trivial.

3 Modeling the System

The increasing complexity of data systems and the workloads they service create challenges for modeling system behavior. We illustrate some challenges with three case studies. The first seeks to predict storage subsystem

(a) 15-dimensional description of storage system workloads in [18]

1. Average arrival rate	4. Percentage of sequential requests
2. Read ratio	5. Temporal and spatial burstiness
3. Average request size	6-15. Correlations between pairs of attributes

(b) 41-dimensional description of directory-level access patterns in [9]

1-3. Number of hours with 1, 2-3, or 4 file opens
4-6. Number of hours with 1-100KB, 100KB-1MB, or >1MB reads
7-9. Number of hours with 1-100KB, 100KB-1MB, or >1MB writes
10-12. Number of hours with 1, 2-3, or 4 metadata requests
13-15. Read request size - 25th, 50th, and 75th percentile of all requests
16-18. Write request size - 25th, 50th, and 75th percentile of all requests
19-21. Avg. time between IO requests - 25th, 50th, and 75th percentile of all request pairs
22-24. Read sequentiality - 25th, 50th, and 75th percentile of files in the subtree
25-27. Write sequentiality - 25th, 50th, and 75th percentile of files in the subtree
28-30. Read:write ratio - 25th, 50th, and 75th percentile of files
31-33. Repeated read ratio - 25th, 50th, and 75th percentile of files
34-36. Overwrite ratio - 25th, 50th, and 75th percentile of files
37. Read sequentiality - aggregated across all files
38. Write sequentiality - aggregated across all files
39. Read:write ratio - aggregated across all files
40. Repeated read ratio - aggregated across all files
41. Overwrite ratio - aggregated across all files

(c) 100,000-plus-dimensional description of datacenter state in [6]

1-100: Machine 1 CPU 25th, 50th, 75th percentiles, memory 25th, 50th, 75th percentiles, ... active threads, page swaps, ...
101-200: Machine 2 CPU 25th, 50th, 75th percentiles, memory 25th, 50th, 75th percentiles, ... active threads, page swaps, ...
...

Table 1: Model complexity for three studies. The number of dimensions corresponds to the complexity of the systems and workloads being modeled

performance at the IO request level and aggregate workload level [18]. The second analyzes two enterprise network storage traces to extract common access patterns, which facilitates identifying highly targeted system optimizations [9]. The third constructs "fingerprints" of datacenter performance crises from historical traces, and uses them to automatically classify future performance crises in real time [6].

The common challenge all these studies confronted is *model multi-dimensionality*. This challenge arises from the simple fact that the systems and workloads being modeled are complex, and therefore need to be described in multiple ways, i.e., multiple dimensions. In other words, the number of dimensions corresponds to the complexity of the systems and workloads being modeled. For example, study [18] models block level storage workloads on disks and disk arrays. The models there used 15 dimensions (Table 1(a)). Study [9] models session, application, file, and directory level storage workloads on client-server network storage systems, an increase in both system and workload complexity. The models there involved up to 41 dimensions (Table 1(b)). Study [6] models general datacenter workloads on the entire datacenter, a further increase in system and workload complexity. The models there involved more than 100 dimensions for each machine, combined across 1000s of machines (Table 1(c)). Multi-dimensionality is *inevitable* for any rigorous attempts to model complex systems and workloads.

Working in multiple dimensions creates a heavy cognitive load, and one instinct is to use as few dimensions as possible, or even resort to heuristics. This approach carries great risk because it introduces *designer bias*, another common challenge. Bias arises from the human system designer's potentially incorrect assumptions and mental models, built up by historical experience, but needs constant re-evaluation as systems and workloads rapidly evolve. Any subjective choice of which dimensions to keep and what heuristical reasoning to apply inevitably involves some assumptions about how systems and workloads behave. All three studies in [18, 9, 6] consciously avoided this approach. They begin with a large number of dimensions, then perform *dimensionality reduction* through classification, clustering, regression, correlation analysis, and other rigorous statistics techniques. Doing so controls the cognitive load in interpreting the models, while minimizing human designer bias.

A further challenge is the necessity to deal with *outlier behavior*. Complex systems servicing complex workloads are prone to introduce rare but regularly appearing outliers. These outliers cause "classical" statistical models to be influenced by rare data. For example, a sudden spike in workload arrival patterns may cause a single system query to take several orders of magnitude longer, causing the arithmetic average query duration to double. For many use cases, this inflated kind of average is not appropriate. The solution is to use *robust statistics*, an active field of statistical research that is gaining in prominence. Some examples of robust statistics include quantiles (general case of percentiles, used in [6]), cluster medians (instead of arithmetic means, used in [9]), and piece-wise models (instead of global models, used in [18]).

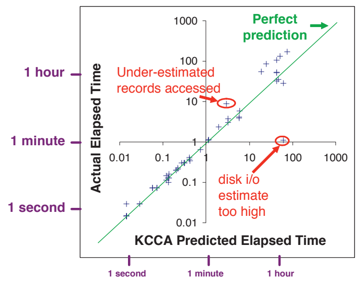
Generally speaking, most well-constructed statistical models can be shown to be effective for some highly specific use cases, even if those models do not adequately address the challenges mentioned here. However, such models create doubt regarding whether they generalize to other use cases, and remain useful as both systems and workloads evolve. The challenges mentioned here require some effort to address, even though solutions to them are more or less known. The rewards of addressing these challenges represent models that survive transient behavior pathologies, are easier to understand and accept, and have a greater likelihood of keeping up with the continuous growth in system and workload complexity.

4 Evaluating a Proposal

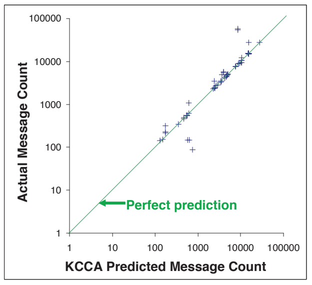
Once the system has been defined and a model chosen, there are two possible next steps. If the research contribution is the system model itself, then the quality of the model needs to be evaluated. Alternately, if the contribution is applying the model to improve system performance, then the evaluation needs to demonstrate both the quality of the model and the improvement in system performance.

An example of evaluating statistical system models is in [13]. The work seeks to develop a multi-dimensional system performance model for database queries. The key statistical tool used is kernel canonical correlation analysis (KCCA). At a high level, KCCA finds dimensions of maximal correlation between an input dataset of query descriptions and an output dataset of query behaviors [5]. Both the input and output datasets are multi-dimensional. Evaluating the system model requires demonstrating that KCCA predicts system behavior that approximates actual system behavior. System behavior traces originate from running an extension of the TPC-DS decision support benchmark [16]. The trace then divides into training and testing datasets, a standard model evaluation technique in statistics. Graphs of predicted versus actual behavior demonstrate the accuracy of the models across multiple dimensions, including query time, message count, and records used (Figure 1).

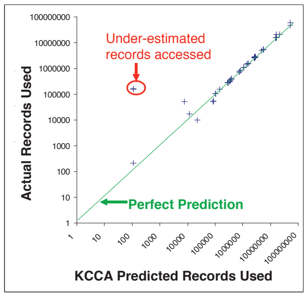
An example of demonstrating improved system performance is in [8]. The work introduces realistic workload suites for MapReduce [10], and uses them to compare the performance of the default MapReduce FIFO task scheduler versus the MapReduce fair scheduler [19]. A fixed workload is replayed under each MapReduce scheduler, and the system performance behavior is observed and compared. In this case, the statistical model is about the input workload, not on the system, and the research contribution is on accurately capturing system behavior subject to realistic workload variations. Such variations complicate performance comparison, in that some conditions favor one system setting, while some other conditions favor other settings. This is true even for simple performance metrics such as job completion time. Figure 2 shows the fair scheduler gives lower job completion time than the FIFO scheduler under some workload arrival patterns, and vice versa under other



(a) Query elapsed time

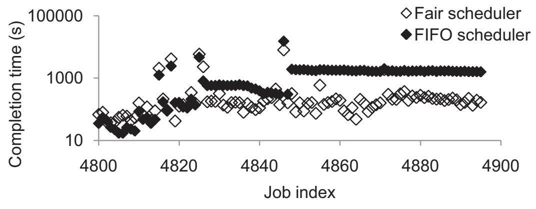


(b) Message count

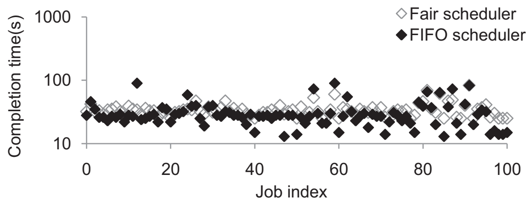


(c) Records used

Figure 1: Evaluating system model - predicted vs. actual behavior. Graphs reproduced from [13].



(a) Fair scheduler has lower job completion time.



(b) FIFO scheduler has lower job completion time.

Figure 2: Evaluating system performance under two settings - MapReduce FIFO scheduler vs. fair scheduler for two different job sequences. Graphs reproduced from [8].

arrival patterns. Thus, the choice of scheduler depends on a rigorous understanding of the workload.

Both of the above studies confront similar challenges. First is the challenge of *workload representativeness*. If the evaluation covers workloads that do not represent real life use cases, then there would be little guidance on how evaluation results translate to real life systems. This challenge is especially relevant for the study in [8], where the choice of the optimal MapReduce scheduler depends on the particular mix of certain arrival patterns. For the study in [13], the standard TPC-DS benchmark is augmented with additional queries that are informed by real life decision support use cases. Such knowledge about real life use cases arises from either empirical analysis of system traces, or from system operator expertise. As data management systems become more complex and more rapidly evolving, it is likely that operator expertise alone would become insufficient, and good *system monitoring* becomes pre-requisite for good system design.

Another challenge is the need for *continuous model re-training*. In [13], both the training and testing datasets come from the trace. This setup does not translate to a real life deployment, in which only the training dataset is available and the test set is generated in real time as queries are submitted to the system. For example, the query prediction model is trained once per configuration and the system takes some action based on the predicted resource requirements for a new query. However, the resulting behavior of the system's response to the new query is not accounted for when the model is static. The concept of a "test dataset" is ad hoc and should be constantly updated by subsequent queries. This is an inherent shortcoming of system behavior models, well-studied in the Internet measurement literature [17]. Consequently, it is desirable to have statistical models that can *constantly re-train parameter values* or even *discover new parameters*.

A third challenge is imposed by the *limitations of system replay*. Replay here implies executing a workload on a real system, with the system making control actions using statistical models. The study in [8] demonstrates this approach. Replay allows designers to explore the interaction between model re-training and system actions that affect the model. However, as data management systems grow in size, replaying long workloads at full scale and full duration becomes a logistical limit. For example, the experiments in [8] required using a 200-

machine cluster for several days. Preparing the experiment required additional days of debugging at scale. Even then, the replay is not at the full scale of the original system that was traced. There is a spectrum of replay fidelity, from comparison experiments on the actual front-line, customer-facing systems, using production-scale data and covering long durations, to scaled-down experiments using artificial data and covering short durations. Ultimately, the system designer needs to judiciously select the *appropriate replay fidelity*, balancing the need for quality insights, logistical feasibility, potential for improvements, and risk of negative system impact.

5 Statistics to Knowledge

The discussion so far has covered the challenges associated with modeling systems and evaluating those models. There is a deeper issue that must be confronted - how to distill knowledge from statistical analysis. Fundamentally, statistical analysis only leads to correlations between workloads, system behavior and performance characteristics. This fact is true even for rigorous hypothesis testing experiments. Statistics help us understand *how* systems behave. There is an inevitable methodological and scientific "leap of faith" to translate from system behavior statistics to some deeper insights about *why* systems behave thus.

To illustrate, consider the study in [12]. This is a follow-up study to [13], and applies the KCCA prediction technique to predict ad-hoc queries running on MapReduce extensions, an increasingly popular complement to traditional databases. The difference in the nature of systems considered in [12] and [13] requires two different model formulations, in that the same KCCA technique need to be applied on two different kinds of system descriptions. While the KCCA technique helps draw statistical correlations between multi-dimensional workload and performance vectors, there is considerable pre-requisite knowledge involved with how to formulate the modeling problem, where to draw the system boundaries, and what actions to take based on the KCCA prediction outputs. Such insights are not supplied by statistical models directly, rather by human designers interpreting the significance of statistical models.

It is non-trivial to interpret the significance of statistical models, especially as systems increase in complexity, and the dimensions of the statistical models likewise grows. Standard dimensionality reduction techniques help somewhat, but ultimately shift the problem from interpreting multiple dimensions to interpreting the dimensionality reduction process.

To illustrate, consider the study in [9], introduced earlier in Section 3. Recall that the study analyzes two enterprise network storage traces to extract common access patterns, which facilitates identifying highly targeted system optimizations. One analysis computes multi-dimensional descriptions of files accessed, and uses k-means clustering [2] to identify the most common types of files. The output of k-means clustering is a set of multi-dimensional cluster centers, as in Table 2. Even though k-means reduced a multi-dimensional space to just six common access patterns, human expertise is needed to interpret the significance of these clusters. Deriving the human applied labels in the lowest row in Table 2 required both examining the numerical values of cluster centers, and computing additional information, such as file types and file co-location within the directory structure. This interpretation step is far more cognitively demanding than applying the k-means algorithm, and arguably represents a transition from a scientific description (statistics) to a scientific taxonomy (knowledge) of system behavior.

More generally speaking, it is desirable to leave multiple opportunities for *human intervention*, both while building statistical models of systems and while using these models to control systems. Human intervention is especially important when multiple performance metrics are involved, and multiple human users have different metrics. The goal of statistical models and statistically managed systems would be more to help mediate potentially conflicting requirements from human users, rather than optimizing for some abstract multi-dimensional performance metric to which the human users cannot relate. Human intervention also offers additional opportunities to detect "black swan" phenomena, which are rare system events that have large performance impact, yet are difficult to statistically identify. A balanced approach would augment statistical methods with human expertise, and vice versa.

File access patterns	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Pattern 6
% of all files	59%	4.0%	4.1%	4.7%	19%	9.2%
# hrs with opens	2hrs	1hr	1hr	1hr	1hr	1hr
Opens per hr	1 open	2-3 opens	2-3 opens	2-3 opens	1 open	1 open
# hrs with reads	0	0	1hr	0	1hr	1hr
Reads per hr	-	-	100KB-1MB	-	1-100KB	1-100KB
# hrs with writes	0	1hr	0	1hr	0	0
Writes per hr	-	100KB-1MB	-	1-100KB	-	-
Read request size	-	-	4-32KB	-	2KB	32KB
Write request size	-	60KB	-	4-22KB	-	-
Read sequentiality	-	-	70%	-	0%	0%
Write sequentiality	-	80%	-	0%	-	-
Read:write ratio	0:0	0:1	1:0	0:1	1:0	1:0
Human applied labels	Metadata only	Sequential write	Sequential read	Small random write	Smallest random read	Small random read

Table 2: Multi-dimensional descriptions of enterprise storage file access patterns from [9].

6 Challenges and Opportunities

Managing data systems using statistical models is becoming an increasingly important topic. The sheer size and complexity of data systems today necessitates some sort of statistical technique to help design and operate such systems. Also, as data systems support essential day-to-day services, it becomes increasingly important to develop accurate behavior and performance models. We believe the broad effort to manage data systems using statistical models remains very much in a nascent stage. We highlight below some opportunities for future research.

One current logistical challenge is the limited availability of large-scale system traces. As discussed in Section 4, the ground truth of constructing and evaluating statistical models arises from monitoring the relevant data systems. However, there are few publicly available traces of large scale, consumer facing systems. This challenge also offers an opportunity, especially for practitioners in industry. The lack of publicly available traces means that organizations who publish their traces stand to influence the direction of the field. Conversely, researchers should keep in mind that the availability of certain kinds of traces does not indicate that those systems are representative. As system tracing capabilities improve and data anonymization tools become prevalent, this challenge can gradually subside.

Another opportunity arises in accelerating the statistically informed data system design loop. Better statistical models lead to better designs and better insights, which in turn lead to better models. As discussed in Section 5, it is non-trivial to make the cognitive advance from each step to the next. One solution would be to identify a small set of robust, scalable, general-purpose statistical tools, with some best-practices and good-defaults on how to apply them, while leaving ample opportunities for human intervention. Another solution would be to devise intuitive visualizations of the modeled statistical behavior. When such visualizations are possible (Figure 2), the human cognitive load vastly decreases. Conversely, it becomes burdensome to interpret multi-dimensional data in a purely numerical fashion (Table 2).

Managing data systems using statistical models requires collaboration between computer system designers and statisticians. An additional challenge lies in the various research incentives for each field. Traditionally speaking, computer system designers get rewarded for building better systems, while statisticians get rewarded for discovering new algorithms or new modeling techniques. The overlap between the two fields is increasing, and both communities are broadening their research agenda. Some examples covered in this paper merely seek to develop a statistical understanding of existing systems instead of constructing new systems. Others merely apply well-known, relatively straightforward statistical methods to assist system design instead of developing new statistical techniques. Practitioners of both fields have much to learn from one another. We are hopeful that the emerging body of work on managing data systems using statistical models helps highlight the growing necessity and rewards for bridging the two fields.

The scale and heterogeneity of today's data systems and the workloads they service complicate system design and operation. As more people are required to build and maintain these systems, statistical techniques have served the systems community well in quantifying the intricacies of data systems. This article discusses some challenges and opportunities that hopefully help inform and guide future studies in the area. We are confident that the confluence of systems and statistics research provides a launching pad for moving the art of system design towards becoming a science.

References

[1] Hadoop World 2011 Speakers. `http://www.hadoopworld.com/speakers/`.

[2] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, Cambridge, Massachusetts, 2004.

[3] Amazon Web Services. Amazon Elastic Computing Cloud. `http://aws.amazon.com/ec2/`.

[4] Apache. Apache Hadoop. `http://hadoop.apache.org/`.

[5] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, March 2003.

[6] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *EuroSys 2010*.

[7] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash, R. Schmidt, and A. Aiyer. Apache Hadoop goes realtime at Facebook. In *SIGMOD 2011*.

[8] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The Case for Evaluating MapReduce Performance Using Workload Suites. In *MASCOTS 2011*.

[9] Y. Chen, K. Srinivasan, G. Goodson, and R. Katz. Design implications for enterprise storage systems via multi-dimensional trace analysis. In *SOSP 2011*.

[10] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[11] EMC and IDC iView. Digital Universe. `http://www.emc.com/leadership/programs/digital-universe.htm`.

[12] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson. Statistics-driven workload modeling for the cloud. In *ICDE Workshops 2010*.

[13] A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *ICDE 2009*.

[14] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg. Quincy: fair scheduling for distributed computing clusters. In *SOSP 2009*.

[15] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: interactive analysis of web-scale datasets. In *VLDB 2010*.

[16] R. O. Nambiar and M. Poess. The making of TPC-DS. In *VLDB 2006*.

[17] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. In *WSC 1997*.

[18] M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, and G. R. Ganger. Storage device performance prediction with cart models. In *MASCOTS 2004*.

[19] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys 2010*.