Towards Integrated Data Center Design

Avrilia Floratou
IBM Almaden Research Center
aflorat@us.ibm.com

Frank Bertsch
University of Wisconsin–Madison
bertsch@cs.wisc.edu

Jignesh M. Patel
University of Wisconsin–Madison
jignesh@cs.wisc.edu

Georgios Laskaris
Duke University
georgios.laskaris@duke.edu

ABSTRACT

Data center design is a tedious and expensive process. Recently, this process has become even more challenging as users of cloud services expect to have guaranteed levels of availability, durability and performance. A new challenge for the service providers is to find the most *cost-effective* data center design and configuration that will accommodate the users' expectations, on ever-changing workloads, and constantly evolving hardware and software components. In this paper, we argue that data center design should become a systematic process. First, it should be done using an integrated approach that takes into account both the hardware and the software interdependencies, and their impact on users' expectations. Second, it should be performed in a "wind tunnel", which uses large-scale simulation to systematically explore the impact of a data center configuration on both the users' and the service providers' requirements. We believe that this is the first step towards *systematic data center design* – an exciting area for future research.

1. INTRODUCTION

Data centers are changing fast. User of data centers running cloud applications are becoming increasingly demanding. These users now expect to have access to specific hardware resources (IOPs, memory capacity, number of CPU cores, network bandwidth), demand data availability and durability guarantees defined quantitatively in Service-Level-Agreements (SLAs) [1,3], and expect to get concrete performance guarantees defined in performance-based SLAs (e.g. [13, 14]). Failure to satisfy these user requirements directly translates into financial loss for the data center/cloud provider. A natural question to ask in this situation is: *"How should we design the data centers of the future?"* In such settings, the data center provider wants to ask various *"What if"* questions, such as what is the cost vs. SLA implication of choosing one type of hard disk over the other, or using flash over hard disk storage, or adding more main memory to each node, or switching to a faster network cards.

Typically, data center designers determine the hardware setup based on the software that will be deployed, and the expected workload characteristics. For example, in an environment where high availability is important, the data is replicated. The value for the replication factor n is typically set to a "reasonable" value (e.g. 3 or 5), and storage is provisioned to accommodate these n copies. However, it is not clear if this approach is the most cost-effective solution that provides the desired availability level. In some environments, one can reduce the replication factor to $n-1$, thereby decreasing the storage cost and increasing the speed of the repair process that re-replicates data in case of a hardware failure. Furthermore, the latency of the repair process can be reduced by using a faster network (hardware), or by optimizing the repair algorithm (software), or both. For example, by instantiating parallel repairs on different machines, one can decrease the probability that the data will become "unavailable" (i.e. the system has zero up-to-date copies of the data). Even if this solution may potentially provide lower availability than n-way replication (depending on the characteristics of the network or the repair algorithm), the resulting availability level may still satisfy the SLA. As we will show in the following sections, similar observations can be made for other design choices too, like data placement algorithms or replication protocols. These observations suggest that an `iterative` approach to data center design, which determines the software configuration first and then the hardware setup (or vice versa), may not be always "optimal" since it ignores important interactions between these two components. What we need is an `integrated` approach that explores the hardware/software interdependencies during the data center design process.

In this paper, we discuss the following three methods for `integrated` data center design: (a) perform what-if analyses on an actual scaled-down prototype of the data center, (b) use purely analytical methods to model the data center, and (c) use a "wind tunnel" to simulate the entire spectrum of hardware/software co-design space for data center design (and incremental re-design/upgrade). We show that the first two approaches have limitations that the third one avoids.

Thus, we envision the creation of a simulation-based data center wind tunnel that can be used to *explore, experiment, and compare* design choices, and to *statistically reason* about the guarantees that these design choices can provide.

This paper is an extended version of the workl in [6]. In the following sections, we present in more detail the three methods for `integrated` data center design and discuss their characteristics. We also discuss the challenges in building

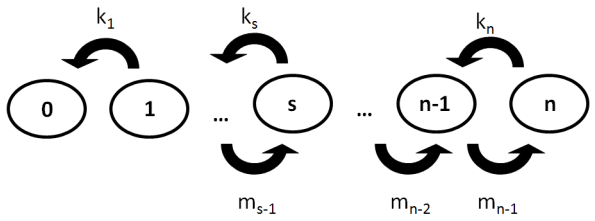

Figure 1: Model for data loss with failures.

a "wind tunnel" for data center design. We believe that these challenges constitute interesting problems that could be addressed by the research community.

2. INTEGRATED DATA CENTER DESIGN

In this section we discuss three approaches to explore hardware-software co-design for data center (DC) design.

2.1 What-if Analysis on a Small Data Center

One approach is to actually experiment with different types of configurations on a small cluster, and perform what-if analyses on the prototype cluster. This approach has several drawbacks.

First, the financial cost to carry out such analyses can be very high, especially if a large number of hardware configurations must be explored, as the number of hardware configurations is a product of the choices that are available for each individual hardware component (e.g. different types of processors, memory, I/O devices, network interfaces, etc.).

Second, this approach requires manual intervention for each hardware setup, so it can be quite expensive from the perspective of the manual effort that is needed.

Third, this approach requires a big time and dollar investment, as the total number of configurations that must be explored is equal to the product of the hardware and the software configurations in the design space.

Finally, it is challenging to pick the "correct" prototype cluster size. The number of nodes in the cluster is a crucial parameter as some behaviors that happen at a larger scale can't be easily observed at a smaller scale. For example, increased communication latency or correlated hardware failures in an actual large scale cluster are harder to model/re-produce in a small prototype cluster.

2.2 Analytical Modeling

Analytical modeling has been widely used to study the behavior of complex systems (e.g., [7]), but such models may result in poor accuracy in their predictions for DC design for the following reasons.

Below, we present an example of a cloud data center design problem that we tried to solve with a Markov chain approach; we then discuss the simplifications that this approach makes, and the limitations of this approach.

In this example, we focus on a scenario in which each customer has n data replicas, distributed across different machines. Our goal is to estimate the probability of permanently losing a customer's data due to failures in the storage layer over a one year period, which we denote as $P(year)$.

We assume that in case of a node failure, one or more repair processes are responsible for replicating the lost data replicas to another node. We also assume that any available replica can be used as a source by only one repair process.

Table 1: Probability of losing data in a one year period.

Replication Factor (n)	MTTF (in hours)	Network Bandwidth	P(year)
1	1M	–	0.87%
1	250K	–	3.44%
2	1M	1 GBit	$2.04 * 10^{-6}$%
2	1M	10 GBit	$2.04 * 10^{-7}$%
2	250K	1 GBit	$3.26 * 10^{-5}$%
2	250K	10 GBit	$3.26 * 10^{-6}$%
3	1M	1 GBit	$7.13 * 10^{-12}$%
3	1M	10 GBit	$6.67 * 10^{-14}$%
3	250K	1 GBit	$4.56 * 10^{-10}$%
3	250K	10 GBit	$4.57 * 10^{-12}$%

The system's behavior can be described as a birth-death process, presented in Figure 1. Each state corresponds to the number of replicas that are currently available. The process is described by birth rates $m_i, 1 \leq i \leq n-1$ and death rates $k_i, 1 \leq i \leq n$. If $k = 1/MTTF$ is the failure rate of the storage subsystem of a machine and $m = 1/MTTR$ is the repair rate, where MTTR (mean time to repair) is the mean time to re-replicate the data stored on a failed machine, then the values of the birth and death rates are:

$$m_i = \begin{cases} m * i & \text{if } 1 \leq i \leq \lfloor n/2 \rfloor \\ m * (n-i) & \text{if } \lfloor n/2 \rfloor + 1 \leq i \leq n-1 \end{cases} \quad (1)$$

$$k_i = k * i, 1 \leq i \leq n \quad (2)$$

If we let Y be the random variable that described the time to data loss (state 0), then by solving the equations produced for the above Markov process, we can compute the cumulative distribution function of Y, denoted as F. The details of the solution are ommitted due to lack of space. The probability that a customer will lose data by time t_d is given by $F(t_d)$. For example, using F, we can easily compute the probability that a customer will lose all the data replicas in a period of one year.

Table 1 shows our results. Here, we show results with a MTTF of 1M hours, which is the typical value that is claimed by the hard disk manufacturers. However, it has been argued that the manufacturers have exaggerated this number, and that the real number should be far lower [16]. Hence, we also show results for an MTTF value of 250K hours. We also assume that the mean time to repair a customer's replica is equal to half of the disk capacity (1TB in this example) divided by the network bandwidth. This is because we expect on average that half of the failed replicas have to be copied before the replica in question is actually repaired.

This example, already shows the usefulness of this analysis style for the service providers that aim to guarantee a certain level of durability to each customer. In fact, analytical models are a very useful tool to *relatively* compare different design choices, as in the example presented above. However, in cloud environments, the accuracy of the model prediction is also very important. This is because every deviance from users' requirements may result in a SLA violation, which translates into financial loss for the service provider. There are two reasons why analytical modeling may result in poor accuracy in their predictions.

First, the (popular) simple M/M/1 and M/M/c models assume that the times to failure and the times to repair are exponentially distributed. However, exponential distributions do not always accurately represent actual distributions encountered in real systems. For example, experimental studies have shown that the hard disk drive failure distribution typically follows the Weibull or Gamma distribution [16]. Moreover, exponential failure distributions are not appropriate for more advanced storage systems (e.g. RAID systems that are comprised of both hard disks and SSDs). Repair times have also been shown to follow the lognormal distribution [17]. The more complex G/G/1 and G/G/C models, which capture more general distributions don't have a closed-form solution. As previous studies have mentioned [12], these more complex models can be approximated but their accuracy is often inadequate.

Second, it is hard to capture all the parameters in the design space with an analytical model. For example, capturing the failure behavior of both the storage and the network layer as well as the repair behavior of the system, using a Markov Model is challenging. Typically the failure behavior of the network components (NIC cards, network switches) is ignored to in order to make the problem tractable. Capturing such complex behaviors accurately with an analytical model is challenging. Capturing the combination of each and every behavior in the design space is even more challenging.

Finally, using analytical models requires deep understanding of the related mathematical concepts, which are not widespread in practice, making it challenging to use this approach in practice. To fix this problem, we hope that more schools will offer courses in quantitative systems modeling – sadly it appears that there are fewer schools that offer these courses in CS departments than a decade ago. We note that analytical models are crucial to better understanding and validating simulation models before the simulator can be used to answer a broader class of what-if questions, and we advocate using analytical models in that role.

2.3 Simulation-based Wind Tunnel

Our proposal to tackle the problem of `integrated` data center design, is to make use of a simulation-based "wind tunnel". We believe that this is the first step towards *systematic data center design* – a rich area of research with potentially high-impact on our industry.

In a wind tunnel the behavior of all the components (hardware, software) under consideration is simulated, with the goal of exploring the behavior of a more complex model. Note that in this case, we are not restricted to exponential distributions as in many analytical models, but we can incorporate any type of distribution.

The simulation-based wind tunnel avoids the hassle of purchasing and repetitively deploying the hardware and software, until we explore all the possible design ideas. Thus, it is a more economical way to investigate DC design choices than using a "sample" DC. At the same time is not restricted, by the size of the "sample" DC.

We note that this philosophy already has some proponents, though in a narrow domain (e.g., [12] for hardware, and [10, 19] for Hadoop parameter selection). What is missing is a general and extensible method that provides a comprehensive framework to evaluate the combination of *both* hardware (storage, CPU, networking and memory components) and software (e.g., replication policy, changes in network priority schemes in SDNs, parameter tuning) design choices. Such a framework is essential to allow holistic and comprehensive evaluation of the entire space of data center design alternatives for end-to-end what-if analyses.

3. WIND TUNNEL USE CASES

In this section we discuss some of the potential DC design problems that the wind tunnel can tackle.

Performance SLAs: To satisfy performance SLAs, workload characterization is needed in order to quantify the impact on existing workloads when a new workload is added on a machine. Existing work in prediction modeling for certain classes of database workloads [13], has shown that it is possible to build accurate models for a given DBMS by identifying and carefully modeling the key characteristics (e.g., CPU, Disk I/O, network, etc.) of the system under test. This observation, encourages us to believe that simulation can be used as a "cheap" way to explore interactions between workloads as long as the key resources are simulated. Our plan is to investigate how much detail the models must capture in order to produce accurate workload predictions and to also validate the predictions on real workloads/environments.

To the best of our knowledge, a performance prediction method that takes into account the impact of other cluster events (e.g., hardware failures, control operations) on workload performance, has not been proposed. Carefully designed, holistic simulation (aka. the "wind tunnel" that we propose) can capture the impact of these events on the performance SLAs, and result in more realistic predictions.

Availability SLAs: Data availability depends on both the hardware (e.g. hardware failure rate) and the software (e.g. replication, erasure codes [15]) characteristics. The wind tunnel is an extensible environment in which various software and hardware system components can be simulated together. Thus, it can be used to experiment with various configurations and collect insights about the impact of each design choice on the availability SLAs. See Section 4 for the challenges associated with modeling these components.

Hardware provisioning: Service providers need to ask questions such as: "Should I invest in storage or memory in order to satisfy the SLAs of 95% of my customers and minimize the total operating cost?" These questions can be described as queries that the wind tunnel can help answer.

4. RESEARCH CHALLENGES

We believe that building a simulation-based wind tunnel is similar to building a specialized data processing system. Thus, the database community is the perfect candidate to address the problem of combining models and data in a wind tunnel for DC design. Queries to the wind tunnel are design questions that iterate over a vast design space of DC configurations. In this section we discuss the research challenges in building the wind tunnel.

4.1 Declarative Simulation Processing

Database researchers have already highlighted the benefits of declarative query processing for computer games [20] and network services [11]. We believe that declarative methods can also help in this setting to capture model interactions and end-user/SLA requirements.

Model interactions: When a new model is added to the simulator, its interactions with the existing models should

be declaratively specified. For example, a model that simulates a data transfer to a machine "conflicts" with a model that simulates a workload executed on this machine. On the other hand, the failure model of the hard disk is independent of the failure model of the network switch. The underlying simulation engine can then automatically optimize and parallelize the query execution based on the user's declarations. Capturing the model interactions declaratively can lead to a modular and extensible wind tunnel design.

Expressing the desired DC constraints: The design questions can often be complicated: for example, the user may need to specify a required SLA as a distribution. Posing such queries declaratively is likely to be preferred over more imperative styles of querying.

We believe, that exploring the design and implementation of declarative languages to support the two aspects discussed above is a valuable area for future research. Perhaps we could build such a language based on existing declarative languages like Datalog or SQL. We may also need two separate languages for each component above – one is a specification language and the other is a query language.

4.2 Simulation at Scale

Simulating multiple components of a DC is likely to be time-consuming. We may need to borrow techniques used to scale database queries (query optimization, parallelization) and apply them in the context of the wind tunnel execution. One approach is to take the following two-step process.

In the first step (optimization), an order for the simulation runs is specified that would facilitate dismissing certain simulation runs by simply observing the output of the previous runs. For example, if a performance SLA cannot be met with a 10GBit network, then it won't be met by having a 1GBit network, while all other design parameters remain the same. Thus, the simulation run with the 10GBit configuration should precede the run with the 1GBit configuration. Extending this idea to more than one dimensions is an interesting research problem, which is similar to multi-way join optimization performed by database optimizers, but with design space configuration parameters and with semantic information associated with the parameters.

In the following step (parallelization), each simulation run is parallelized based on the model interactions (perhaps specified declaratively as discussed above). For example, the affected components during the completion of a data transfer from one node in a rack to another node in the same rack are the two nodes, the two disks where the data was streaming to and from, and the switch itself. Work done on other nodes within the rack is unaffected, and work done between any nodes not in the rack remains unaffected as well. Parallel execution of events such as this provides an easy way to scale the simulator. Although, the problem of parallelizing simulations has been studied before (e.g., [8]), declarative query processing creates new opportunities and challenges. The logic specific to the models being simulated must now be captured when declaring the models and not in the simulation engine itself. As a result, the simulation engine should be designed to gracefully adjust to various user declarations. Exploiting whether the existing simulation parallelization techniques can be abstracted in a way that allows coupling with a declarative specification language is an interesting direction for future work.

Another approach to speed up execution, is to monitor the simulation progress and abort a simulation run before it completes, if it is clear from the existing progress that the design constraint (e.g. a desired SLA) will not be met. Exploiting this knowledge to refine the order of future configurations is another interesting problem, and similar to dynamic query optimization.

4.3 Validating the Simulator

Another challenge is how to validate the simulator. Simple simulations can be validated using analytical models. The predictions of crucial models can be tested on a small hardware setup, when ever that is possible. Another way to validate the simulation results, is to make use of publicly available datasets that describe the behavior of real clusters. There are a few datasets of this kind that have been published (e.g., [16]), but we certainly need more data to be able to study in depth all the problems related to DC design. In fact, one hope that we have is that a broader recognition of this area of systematic DC design, will enable researchers to invest time in producing datasets from actual deployments, that can then be used in wind tunnel simulations. Perhaps, projects like OpenCompute [2], could expand from their current charter to take on the challenge of making data from large operational systems available on a continual basis, especially for newer hardware components. It would also be useful to get sanitized hardware and software logs from actual operational clusters to allow validation of selected simulations (e.g., response time prediction).

4.4 Managing the Wind Tunnel Data

The data related to the wind tunnel falls into two categories: (a) data generated by wind tunnel simulations, and (b) operational log data from DCs used to build data-driven models for the simulator components.

Akin the situation with simulation data in the sciences, we expect that a large amount of simulation data (generated by the process of exploring over large portions of the DC design space) will be collected over time. This data can be subjected to deep exploratory analysis. For example, users may question whether they have already explored a configuration scenario "similar" to a target scenario, or question what is a typical pattern of failures for an specific configuration(s). Determining how to organize the output of the simulation data along with the input model data (e.g. should we use a relational store or a schema-less key-value store?) to facilitate these kinds of analyses is an interesting problem. The varied nature of this data makes this problem challenging [4, 9].

Identifying how to store and process operational log data from real DCs in order to seed data-driven models (e.g. for an SSD device type) is an interesting problem. As mentioned in [9], transformation algorithms that convert log data into meaningful models (e.g., probability distributions) that can be used by the wind tunnel, must be developed. The related data privacy issues must also be addressed.

4.5 Modeling the Hardware Components

Satisfying availability and durability SLAs, requires precise knowledge of the failure characteristics of each hardware component. For example, modeling the failure behavior of a hard disk, requires a good estimate of the distribution that time between disk replacements follows. Although, hardware manufacturer's typically enclose the values of the mean

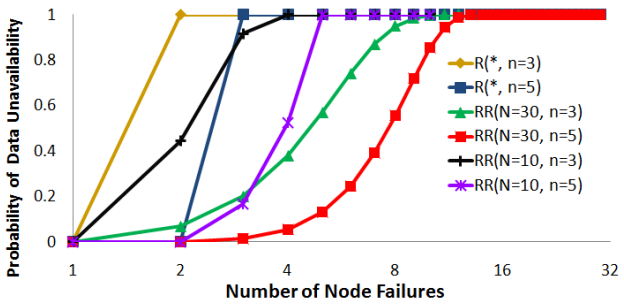

Figure 2: Probability of data unavailability

time to failure and the annualized failure rate, these numbers do not necessarily reflect the actual disk behavior. In fact, these numbers tend to overestimate the hardware capabilities [16]. Over the last few years, more hardware failure studies have been conducted [17, 18], but the entire space of hardware components (such as I/O controllers, memory modules, etc), has still not been covered.

Another problem often encountered in large DCs is hardware whose performance deteriorates significantly compared to its specification [5]. This kind of behavior (e.g, an underperforming NIC card) are hard to reproduce in practice. We believe that simulation is an excellent way to study the effects of these behaviors on the DC operations. Examining how to model these "failures" across the hardware components spectrum is an exciting area of future research.

4.6 Modeling the Software Components

Besides better hardware models, we also need models for software components. One key challenge in modeling software is to decide which software design choices affect the SLA requirements, and as a result should be simulated.

As an example, we present early results from a simulator that we are building. Here we simulate a cloud setting in which data is replicated for high availability. Figure 2 shows the probability of having at least one customer's data become unavailable as the number of node failures in the cluster increases, for varying cluster sizes, data placement algorithms and replication factors. We assume that the service operates using a quorum-based protocol. In this case, if the majority of data replicas of a given customer becomes unavailable, the customer is not able to operate on the data. We use the *Random (R)* and *Round Robin (RR)* data placement policies to distribute the replicas across the machines. We use $10,000$ users, two replication factors ($n = 3$, $n = 5$) and two configurations (N = 10 and N = 30 nodes). The * symbol denotes both cluster configurations. As the figure shows, the probability of data unavailability varies across different configurations. It depends on the cluster size, the replication factor and the data placement policy. Thus, all of these parameters must be captured in the simulation in order to quantify the system's availability behavior.

Determining which software design choices interact, and as a whole affect a specific aspect of the system's behavior, such as data availability, is challenging. Failure to model a critical parameter may result in inaccurate simulation results. An interesting research problem is to systematically analyze existing complex software systems and determining which parameters are independent from the others and which ones interact with others. Although, this categorization is needed for accurate DC design with the wind tunnel, the categorization can be made easier by making use of the wind tunnel itself! This is because the wind tunnel constitutes a framework in which we can add or remove properties and get insights on the effect that these produce at a global scale – thus the wind tunnel can help software engineers make better design choices.

5. CONCLUSIONS

It is too expensive and sub-optimal to design data centers using existing methods. As the diversity of the hardware components ecosystem (e.g., new storage device types, low powered processors, network speeds) continues to increase, along with an explosion in software methods (e.g., replication, data partitioning, distributed query processing), it is critical that we investigate simulation-based methods to answer holistic "what-if" questions on simulated data center designs. We proposed building a wind tunnel to address this problem, and outline various research challenges that must be addressed to realize this vision.

6. REFERENCES

[1] Amazon ec2 sla. `http://aws.amazon.com/ec2-sla/`.

[2] Open compute project. `http://www.opencompute.org`.

[3] Windows azure slas. `http://www.windowsazure.com/en-us/support/legal/sla/`.

[4] P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. A vision for management of complex models. *SIGMOD Rec.*, 29(4):55–63, Dec. 2000.

[5] T. Do, M. Hao, T. Leesatapornwongsa, T. Patana-anake, and H. S. Gunawi. Limplock: Understanding the impact of limpware on scale-out cloud systems. In *SOCC*, 2013.

[6] A. Floratou, F. Bertsch, J. M. Patel, and G. Laskaris. Towards building wind tunnels for data center design. *PVLDB*, 7(9):781–784, 2014.

[7] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *OSDI*, pages 61–74, 2010.

[8] R. Fujimoto. Parallel discrete event simulation. *Commun. ACM*, 33(10):30–53, 1990.

[9] P. J. Haas, P. P. Maglio, P. G. Selinger, and W. C. Tan. Data is dead... without what-if models. *PVLDB*, 4(12):1486–1489, 2011.

[10] H. Herodotou and S. Babu. A what-if engine for cost-based mapreduce optimization. *IEEE Data Eng. Bull.*, 36(1):5–14, 2013.

[11] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking. *Commun. ACM*, 52(11):87–95, Nov. 2009.

[12] D. Meisner, J. Wu, and T. F. Wenisch. Bighouse: A simulation infrastructure for data center systems. In *ISPASS*, pages 35–45, 2012.

[13] B. Mozafari, C. Curino, and S. Madden. Dbseer: Resource and performance prediction for building a next generation database cloud. In *CIDR*, 2013.

[14] V. R. Narasayya, S. Das, M. Syamala, B. Chandramouli, and S. Chaudhuri. Sqlvm: Performance isolation in multi-tenant relational database-as-a-service. In *CIDR*, 2013.

[15] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. Xoring elephants: Novel erasure codes for big data. *Proc. VLDB Endow.*, 6(5):325–336, Mar. 2013.

[16] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you? In *FAST*, pages 1–16, 2007.

[17] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. *IEEE Trans. Dependable Sec. Comput.*, 7(4):337–351, 2010.

[18] K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *SoCC*, pages 193–204, 2010.

[19] G. Wang, A. R. Butt, P. Pandey, and K. Gupta. A simulation approach to evaluating design decisions in mapreduce setups. In *MASCOTS*, pages 1–11, 2009.

[20] W. White, C. Koch, N. Gupta, J. Gehrke, and A. Demers. Database research opportunities in computer games. *SIGMOD Rec.*, 36(3):7–13, Sept. 2007.