Lessons from Teaching Analytical Performance Modeling

Y.C. Tay
dcstayyc@nus.edu.sg
National University of Singapore

ABSTRACT

This talk summarizes some lessons from teaching a course in analytical performance modeling, specifically: (1) what can an analytical model offer? (2) how a model can be decomposed into submodels, so as to decouple different forces affecting performance, and thus analyze their interaction; (3) making choices in formulating a model; (4) the role of assumptions; (5) Average Value Approximation (AVA); (6) when bottleneck analysis suffices; (7) reducing the parameter space; (8) the concept of analytic validation; and (9) analysis with an analytical model.

CCS CONCEPTS

• **General and reference** → **Performance**; • **Information systems** → **Database performance evaluation**; • **Networks** → **Network performance modeling**; • **Software and its engineering** → **Software performance**; • **Computing methodologies** → *Modeling and simulation.*

KEYWORDS

performance; analytical model; approximations; assumptions; analysis; validation

ACM Reference Format:
Y.C. Tay. 2019. Lessons from Teaching Analytical Performance Modeling. In *Tenth ACM/SPEC International Conference on Performance Engineering Companion (ICPE '19 Companion), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3302541.3311527

1 INTRODUCTION

After teaching analytical performance modeling for some years, I published my lecture notes as a book [1]. One target user of the book is the engineer who (a) is interested in the performance of a particular system and (b) wants to model its behavior analytically, but (c) does not intend to become an expert in mathematical modeling. The techniques in the book are illustrated with 40 papers chosen from a broad range of computer systems, big and small.

[1] *Analytical Performance Modeling for Computer Systems*, 3rd Edition, Morgan & Claypool, 2018

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPE '19 Companion, April 7–11, 2019, Mumbai, India
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6286-3/19/04…$15.00
https://doi.org/10.1145/3302541.3311527

2 LESSONS

In the following, I summarize the major lessons in the book, using some of the papers as examples.

2.1 Why an analytical model?

Cloud providers offer their customers an attractive proposition that the resources provided can be elastically scaled according to demand. However, a tenant may need a tool to help determine how much compute power, memory, etc. to acquire, but is hampered by having very little knowledge of the cloud architecture. **[Elastic-Scaling]** [3] shows how one can get around this difficulty with a model.

Energy and latency are major issues for datacenters. **[DatacenterAMP]** [7] uses a simple queueing model to study how asymmetric multicore processors (AMP) can help the tradeoff between energy consumption and latency bounds. The idea is to marshal the resources of smaller, less power-intensive cores into one large, faster core. The technology for this is not yet ready, so a model is necessary for exploring what is (or is not) possible in reconfiguring cores for performance.

[GPU] [9] also uses a model for architectural exploration. A GPU runs multiple threads simultaneously. This speeds up the execution, but also causes delays from competition for MSHRs (Miss Status Handling Registers) and the DRAM bus. The paper uses a model to analyze how changing the number of MSHRs, cache line size, miss latency, DRAM bandwidth, etc. affect the tradeoff. There is no practical way of doing such an exploration with real hardware. Part of the analysis is based on using the model to generate the CPI (cycles per instruction) stack that classifies the stall cycles into MSHR queueing delay, DRAM access latency, L1 cache hits, etc.

The model in **[SoftErrors]** [13] is similarly used to determine how various microarchitectural structures (reorder buffer, issue queue, etc.) contribute to the soft error rate that is induced by cosmic radiation. Again, tweaking hardware structures is infeasible, and simulating soft errors is intractable, so an analysis with a mathematical model is essential.

One equation from the **[SoftErrors]** model explains why an intuition — that a workload with a low CPI should have a low vulnerability to soft errors — can be contradicted. Similarly, the CPI stack breakdown by the **[GPU]** model explains how workload that spends many cycles queueing for DRAM bandwidth can, counter-intuitively, have negligible stall cycles from L2 cache misses.

To design a complicated system, an engineer needs help from intuition that is distilled from experience. However, experience with real systems is limited by availability and configuration history. Although one can get around that via simulation, the overwhelming size of the parameter space usually requires a limited exploration; this exploration is, in turn, guided by intuition. One way of breaking this circularity is to construct an analytical model that abstracts

away the technical details and zooms in on the factors affecting the central issues. We can then check our intuition with an analysis of the model.

Intuition is improved through contradictions: they point out limits on the old intuition, and new intuition is gained to replace the contradictions. Again, such contradictions can be hard to find in a large simulation space, but may be plain to see with the equations in a mathematical model.

2.2 Decomposition and decoupling

In the **[SoftErrors]** equation that explains why an execution with low CPI can have a high vulnerability to soft errors, critical path length K and instruction latency ℓ appear in a product. K is determined by the workload and ℓ by the microarchitecture; in this way, the model decouples the two main factors (workload and microarchitecture) that determine performance, so one can analyze their impact through K and ℓ separately.

For **[ElasticScaling]**, the context is an in-memory transactional data grid (e.g. a NoSQL data store). The performance is determined by an interaction between contention among transactions for access to data, and competition for cloud resources. The paper decomposes this interaction into two submodels, one for data contention and another for resource contention. The submodel for data contention is a *whitebox* that explicitly models the access pattern, contention for data locks, etc.; in contrast, the cloud architecture is largely unknown to the tenant, so the submodel for resource contention is a *blackbox* that uses machine learning to predict delays. Interaction between data and resource contention is modeled by iteration between solutions from the two submodels.

How might the two forms of contention affect one another? One can see this in **[DatabaseSerializability]** [2], which considers an application (e.g. e-commerce) that runs on a system that has a master database and a middle-tier caching layer that is distributed. The model is used to study how much performance one can buy for a transaction, by letting it read data from a stale cache. There, an increase in data contention (e.g. more concurrent transactions) causes more transactions to wait for locks, thus reducing resource contention; on the other hand, the increased blocking causes more freshness violation, and thus more aborted transactions and wasted resources. Conversely, an increase in the number of caches makes it more likely that a transaction can access a local copy, possibly reducing data contention; however, more caches also implies more effort to enforce their consistency, and that may lead to higher data contention. In contrast to **[ElasticScaling]**, the resource contention here is modeled with a whitebox, with object queues, caches, etc.

The jobs in a system may suffer delays for various reasons. In **[MapReduce]** [21], a job consists of map, shuffle and reduce tasks that run in a distributed system. The job performance depends not only on the queueing delays for resources (cycles, disks, etc.), but also on the precedence constraints among the tasks. Again, the model uses two submodels, a standard one for the resource contention, and another one for the precedence constraint. The precedence submodel is used to determine the number of active tasks that compete for resources in the other submodel, thus separating the two major factors that determine job performance.

2.3 Making choices

One precedence constraint in **[MapReduce]** is the synchronization of map tasks. The model needs to estimate the average of the maximum task times, i.e. $E\max\{X_1, \ldots, X_n\}$. The authors first did some experiments to verify that this average cannot be accurately approximated by $\max\{EX_1, \ldots, EX_n\}$. One usually think of experiments as coming after a model is done, to generate measurements for validation. The **[MapReduce]** example thus shows that experiments are sometimes also needed to guide the formulation of the model.

Results from a model can also guide the implementation of a system, so the two could be developed with help from each other. This is the case for **[P2PVoD]** [5]. The runaway success of P2P systems like BitTorrent has prompted many proposals from academia to improve such protocols. Each proposal can be viewed as a point in the design space. This paper proposes an analytical model to represent the entire space defined by the three dimensions of throughput, sequentiality and robustness. The tradeoffs in choosing one point instead of another in this space can then be analyzed via the model.

Without guidance from experiments, one may make the wrong choices, and possibly end up with a model that is unnecessarily complicated.

On-chip interconnects are replacing buses and point-to-point wires. The performance, fault-tolerance, and energy issues for such network-on-chip are inter-related. For example, errors from crosstalk, electromagnetic interference and cosmic radiation (of increasing importance with miniaturization) can cause packet retransmissions, thus increasing latency and energy consumption. **[NoC]** [10] proposes a queueing model as a design tool for fast evaluation of router designs. The model's estimate of power consumption is close to simulation measurement, but it is also roughly linear with respect to flit injection probability. This linearity means that some of the nonlinear expressions in the model could be simplified to give a simpler model that might bring clearer insight.

Running experiments first can also prevent one from choosing a modeling technique that is simpler than necessary. The queueing network model in **[PipelineParallelism]** [14] is used to study how a parallel computation should be balanced over a pipeline of stages, with concurrent threads per stage, so that slow execution in one stage does not stall the other stages. Experiments show execution time is not monotonic with respect to the number of threads; standard queueing networks do not have such nonmonotonic behavior, so the modeling technique chosen is inadequate.

[WirelessCapacity] [6] presents a different choice, between two models. The system consists of stationary wireless nodes that are geographically distributed, and traffic between nodes may require multiple hops. The channel is shared, and senders contend for medium access. The issue is scalability: How does the number of nodes affect the achievable throughput per sender? The paper presents two models, one at the physical layer, and one at the protocol layer. They give different upper bounds on the transport capacity, but there are no experiments in the paper to help decide which is better.

2.4 The role of assumptions

The absence of experimental validation in **[WirelessCapacity]** is perhaps understandable: A real network is hostage to the flakiness of wireless interference and hard to scale up, whereas simulators make simplistic assumptions (e.g. interference and mobility patterns).

Analytical models are often criticized and dismissed for their simplifying assumptions. Indeed, one must take care that a model's assumptions and approximations do not introduce artifacts, nor miss essential properties of the system.

In **[SensorNet]** [18], the system consists of stationary wireless sensors, and some mobile data mules that collect their measurements for delivery to access points. The basic issues are how buffer sizes should scale and how data delivery rates would change with the system size. The model discretizes the space into N locations, and derives a buffer occupancy that depends on N. This result is clearly an artifact of the model.

On the other hand, the assumptions in **[PipelineParallelism]** could not model the nonmonotonicity in execution time.

[GPRS] [15] offers an example that can model nonmonotonic behavior. The GPRS infrastructure has multiple overlapping wireless cells managed by base stations wired to a backbone of routers, and this paper focuses on the timeslot allocation for downlink traffic to mobile nodes in one bottleneck cell. For a fixed number of users per cell and as the load increases, the model shows that throughput first increases, then decreases.

Another example is **[PerformanceAssurance]** [17], which examines resource provisioning for multi-tier services at the granularity of software components. It has an analytical model that uses an MVA algorithm from queueing theory to compute performance estimates, and these are used to decide how components are to be replicated and distributed to satisfy service level objectives. MVA assumes service demands do not change with queue size, but this is not the case here, because of context switching, etc. Instead of worrying about this assumption, the authors violate it by modeling the change in service demands, and feed them to the MVA algorithm anyway. The resulting estimates show a nonmonotonic behavior in CPU utilization that matches experimental measurements.

The model in **[DatabaseScalability]** [4] also uses the MVA algorithm despite violating the MVA assumptions. The paper illustrates how a model can use the performance of a standalone database to predict the performance of a replicated database and, in addition, compare two alternative designs (multi-master and single-master). The probability of aborting a transaction increases with the replication and the number of clients, thus changing the demand for resources (at the CPU, disks, etc.) and breaking the assumptions underlying MVA. Yet, experiments show good agreement between model predictions and simulation measurements.

[MapReduce] similarly break the MVA assumptions in using the precedence constraints to adjust the number of concurrent tasks in the MVA algorithm.

In **[GPU]** [9], the model puts a bound on the delay computed with the Pollaczek-Khinchin formula for an $M/D/1$ queue. One might consider this as a violation of the assumptions for the formula, but another way to see it is that the bound merely modifies an approximation. After all, the $M/D/1$ queue is itself an approximate model of DRAM bandwidth contention delay.

Some models work very hard to generalize the assumptions for the derivations. Such efforts are unnecessary — it serves no purpose to precisely determine the assumptions (e.g., Markov regenerative process) that are needed to justify an equality when so much else in the model are approximations. They only transfer the need for justification from the equation to the assumptions. One should just regard the equality as another approximation.

Besides, the objection against simplifying assumptions is partly based on a confusion between sufficient and necessary conditions for a derivation. The assumptions may suffice for the results, but that does not mean that they are necessary. Often, the results from the analysis are robust with respect to violations of the assumptions, and continue to hold even under weaker assumptions; the simplifying assumptions are then a quick way of arriving at the results.

One should therefore not hesitate to adopt simplifying assumptions, push on with the analysis, and let the experimental measurements justify the derivation.

2.5 Average Value Approximation (AVA)

In line with simplifying assumptions, one technique that is often used in performance modeling is to replace a random variable X by its mean EX. I call this **Average Value Approximation (AVA)**.

For example, the **[SoftErrors]** model estimates a critical path C as taking ℓK cycles, where ℓ is average instruction latency and K is the average critical path length. In other words, it adopts the approximation $E\tilde{C} = E(\tilde{\ell}\tilde{K}) \approx (E\tilde{\ell})(E\tilde{K})$, where $\tilde{C}$, $\tilde{\ell}$ and $\tilde{K}$ are the respective random variables. A sufficient condition for this derivation to hold is if $\tilde{\ell}$ and $\tilde{K}$ are independent, but this may not be true for the workloads in **[SoftErrors]**. The approximation is thus an example of AVA.

Similarly, in **[GPRS]**, the model uses an approximation $E\tilde{X} = E\frac{\tilde{U}}{\tilde{Q}} \approx \frac{E\tilde{U}}{E\tilde{Q}}$, where $\tilde{X}$ is the number of time slots given to a mobile for its packet transfer, $\tilde{U}$ is the number of time slots occupied by active mobiles, and $\tilde{Q}$ is the number of mobiles in active transfer. This is also an example of AVA since, strictly speaking, $E\frac{X}{Y} = \frac{EX}{EY}$ may not hold even for independent random variables X and Y.

More examples of AVA can be found in **[ElasticScaling]**, **[SensorNet]**, and other papers described below.

2.6 Bottleneck analysis

No matter how complex a system is, estimating its performance is usually easy at two extremes: when workload is light and no time is wasted on queueing for resources, and when workload is heavy and performance is determined by a bottleneck resource that is rarely idle. These extremes determine two straightline asymptotes that meet at a *knee*, and performance is a nonlinear curve around this knee, but converges to these asymptotes for light and heavy workloads. Sometimes, it suffices that the model locate these two straight lines via a bottleneck analysis.

[Roofline] [22] provides an example, where the aim is to determine whether memory bandwidth or processor speed is the bottleneck for a workload. With just the two asymptotes, one can

choose between two architectures by comparing their knees, evaluate the efficacy of a software optimization by considering how it affects the lines, and determine if an application is constrained by memory bandwidth or processor speed.

For a larger example, consider **[CloudTransactions]** [11]. Cloud providers use elasticity as one selling point for their services. These providers offer a choice of architectures, and this paper's objective is to compare the scalability and cost of three choices: *classic* (e.g. Amazon Relational Database Service), *replicated* (e.g. Microsoft Azure) and *partitioned* (e.g. Google AppEngine). A comparison can be done via a bottleneck analysis that examines how the asymptotes are affected by the number of servers (web/application, database and storage), service demands and arrival rate of transactions and ratio of server speeds.

The conclusions in **[MapReduce]** are based on configurations where the response time is mostly linear and utilization is flat with respect to the number of map tasks, i.e. performance has hit a bottleneck. That says that a simple bottleneck analysis should suffice, without need for the difficult integration of MVA and precedence constraints.

Similarly, the throughput and response time in **[DatabaseScalability]** are mostly linear with respect to the number of replicas (which determines the number of clients), so a bottleneck analysis should suffice to draw many of the conclusions, without need for a transaction conflict analysis.

2.7 Reducing the parameter space

A system can have many parameters of interest. One advantage that an analytical model has over measurements with implementations or simulations is the ability to look at the entire parameter space. For example, the **[P2PVoD]** model explains a counter-intuitive behavior and reflects a performance tradeoff as one moves from one part of the parameter space to another.

On a related note, **[DatabaseSerializability]** shows how one concurrency control has better throughput than another for one part of the parameter space, and vice versa for another. This illustrates how two simulation studies can reach different conclusions if the parameters are set differently, using some "magic" values.

A bad choice of magic values can mislead an analysis. For example, some experiments in **[PipelineParallelism]** show that it is optimal to collapse all intermediate stages of the computation pipeline; in that case, there is no need for an analytical model to decide how to structure the pipeline and threads. This conclusion suggests that the analysis may be looking at some uninteresting region of the parameter space.

Instead of magic values, one better way of reducing the parameter space is to restrict the analysis to realistic scenarios. For **[GPRS]**, it is considered unacceptable for the probability of blocking (i.e. a mobile that wants to start a download cannot do it because the number of active mobiles in the cell has reached a maximum) to exceed 2%. Yet, some experiments push the blocking probability to levels far exceeding that. By using that 2% to restrict the parameter space, the analysis could possibly be simplified, and yield clearer insight.

[EpidemicRouting] [23] illustrates another way of restricting the parameter space. The system consists of wireless nodes moving in some pattern over a restricted area. Each node is a source and a destination for a flow of packets, has a buffer to hold packets, and runs a protocol to route a packet from a source to its destination via forwarding when two mobile nodes meet wirelessly. This forwarding is akin to infection except, being a protocol, there are multiple variations to the infection mechanism. One objective of this paper is to propose a uniform framework, using differential equations, for modeling the variations. The paper avoids the difficult issue of modeling the mobility pattern by drawing on previous work to reduce that problem to a parameter β that represents the pairwise node meeting rate. The analysis then proceeds with β, without a need to consider the details of node movement. This β in fact aggregates multiple parameters — the size of the restricted area, the average relative speed between two nodes, etc.

[TransactionalMemory] [8] gives another example for the use of an aggregated parameter. Single-processor techniques for coordinating access to shared memory (e.g. using semaphores) do not scale well as the number of cores increases. One possible alternative is transactional memory, and this paper studies the performance impact when it is implemented in software. The main parameters for the model are the number of data items L, the number of concurrent transactions N, the number of lock requests k, and the probability that a lock request is for write access. Part of the analysis uses the aggregated parameter kN/L to delimit their parameter space.

In **[MediaStreaming]** [20], the system has a set of streaming servers and peers that are connected by a network, and a collection of files for download. The scalability of the system relies on harnessing the bandwidth of peers, by having them upload their downloaded files to other peers. One objective is to determine the time k_0 it takes for the peers to take over from the servers, in terms of satisfying download demand. The model uses a transient analysis to derive $k_0 = (\log W)/\log(1+C)$, where $W = \lambda Lb/N$, $C = \alpha/b$ and α, λ, b, L and N are parameters for peer and server bandwidth, bandwidth to stream a file, length of a streaming session and request rate. We see here that one can easily explore the 2-dimensional space for k_0 defined by C and W, instead of a 5-dimensional space.

In fact, an aggregated parameter can encapsulate a space of unknown dimensions. One major difficulty for **[ElasticScaling]** lies in modeling the transactions' nonuniform and unknown data access pattern. The authors conjecture that this nonuniform access is in fact equivalent to uniform access over a dataset of size $D = 1/\text{ACF}$, where the Application Contention Factor ACF can be calibrated (and the conjecture validated). It would take an unknown number of parameters to exactly specify the nonuniform access pattern, so this ACF simplifies the analysis tremendously.

2.8 Analytic validation

Analytic modeling is an art, in that we pick approximations to trade accuracy for tractability and insight. A different choice of approximations would give different expressions for the performance measures. Simplifying assumptions can also introduce artifacts that are not properties of the system.

Hence, theoretical conclusions from an analytical model must be validated to make sure that they describe properties of the system, rather than properties of the model. I call this **analytic validation**.

Instead of comparing experimental measurement and model calculation, one can do an analytic validation by comparing experimental measurements. To give an example, recall the result $k_0 = (\log(\lambda Lb/N))/\log(1 + (\alpha/b))$ from the **[MediaStreaming]** model. We can check if this is true of the system (and not just in the model), as follows: Let $X = k_0 \log(1 + (\alpha/b))$ and $Y = \log(\lambda Lb/N)$, *measure* k_0 for a choice of α, b, etc., repeat this experiment r times to get $\langle X_1, Y_1\rangle, \ldots, \langle X_r, Y_r\rangle$, and plot these r points on the X-Y plane; if the system in fact has the property derived with the model, then the points should cluster around the $X = Y$ line.

For another example, recall the conjecture in **[ElasticScaling]** that the nonuniform access of real transactions is equivalent to uniform access over a dataset of size 1/ACF. The key property to check her is whether the *measured ACF* is in fact constant for a workload. Experimental results in the paper show that, in fact, this is the case as the number of nodes in the data grid increases, and despite a drastic change in application workload (TPC-C and Radargun) and system architecture (private cluster and EC2).

Internet routers drop packets when their buffers are full. To avoid this, and to accommodate the large number of flows, router buffers have become very big. **[RouterBuffer]** [1] uses an analytical model to study whether flow multiplexing can make this bufferbloat unnecessary. Among the results are expressions for buffer occupancy EQ and $\text{Prob}(Q \geq B)$ that depend on the link rate C only through link utilization ρ. These are strong claims from the model, so they require experimental verification. Indeed, simulation results in the paper show that when ρ is fixed, EQ for different values of C are indistinguishable, and similarly for $\text{Prob}(Q \geq B)$. These confirm that the "depends on C only through ρ" result is a property of the system, and not an artifact of the model.

In wireless networking, WiFi technology is based on the IEEE 802.11 protocol that controls packet exchange between the base station and mobiles in the cell. Simultaneous transmissions from different mobiles can cause packet collisions and induce retransmissions and backoff, so maximum possible throughput can be lower than the channel bandwidth. The model in **[802.11]** [19] examines how this saturation throughput depends on the protocol parameters, and on the tradeoff between collision and backoff. One of the claim from the model says that the probability p of a collision depends on the protocol's minimum window size W and the number of mobiles n only through the gap $g = W/(n-1)$. Simulations show that $\langle g, p\rangle$ for different configurations do in fact lie on the same curve. The model also claims that bandwidth wasted by collisions exceeds idle bandwidth caused by backoffs if and only if $r > 1/T$, where r is the transmission rate and T is the transmission time (including headers); this was also analytically validated by the experiments.

2.9 Analysis with an analytical model

To answer the question "Why an analytical model?", we had earlier looked at specific examples of what such a model can offer, in the case of **[ElasticScaling]**, **[DatacenterAMP]**, **[GPU]** and **[SoftErrors]**. But there is another answer, from the perspective of Computer Science.

A common argument for an analytical model is that solving it is faster than simulations and, in fact, many papers use their models as replacement for simulators: They are used to generate numerical results that are plotted to show the relationship between performance measures and input parameters, and conclusions are drawn from looking at the plots. I call this **analytic simulation**.

If conclusions are to be drawn from looking at plots, then one may be better off taking the time to generate those plots with a simulator, since the model's assumptions and approximations may introduce misleading inaccuracies. The power in an analytical model lies not in its role as a fast substitute for a simulator, but in the analysis that one can bring to bear on its equations. Such an analysis can yield insights that cannot be obtained by eyeballing any number of plots (without knowing what you are looking for), and provide conclusions that no simulator can offer.

There is a related argument that an analytical model can be used as a tool to help system design (e.g. **[NoC]**, **[DatabaseScalability]**) or resource provisioning (e.g. **[GPRS]**, **[PerformanceAssurance]**). However, a model can offer more than that.

The transport protocol TCP is a software that controls much of current Internet traffic. Most TCP versions use packet loss and timeout as signals for controlling the transmission rate, so a natural question is: how do they affect the connection throughput? The protocol has intricate details, and runs over a distributed mess of hardware, yet the model in **[TCP]** [16] is able to capture the essence of TCP behavior. Its key equation expresses TCP throughput in terms of loss probability p and round trip time RTT. Clearly, for any nontrivial Internet path, p and RTT can only be measured, not predicted, so what is the point of having that equation? Its significance lies not in predicting throughput, but in characterizing its relationship with p and RTT. Such a characterization led to the concept of TCP-friendliness, and the design of equation-based protocols. The model thus advances the science of network communication.

For a topical example, consider the spread of fake news and memes, etc. over the Web, driven by user interest, modulated by daily and weekly cycles, and dampened over time. To adequately capture this behavior, **[InformationDiffusion]** [12] modifies the classical epidemic model. However, there is no way of integrating the resulting differential equation, so it is solved numerically. The parameters can then be calibrated by fitting measured data points. Since measurements are needed for calibration, the model has limited predictive power. Nonetheless, the parametric values serve to succinctly characterize the diffusion, and provide some insight into its origin. This example illustrates the point that, although the pieces of a system are designed, engineered and artificial, they can exhibit a hard-to-understand, organic behavior when put together. An analytical model is thus a tool for developing the science of such organisms.

There are other examples of how an analytical model can contribute to the scientific analysis of a computer system: the tradeoff between collision and backoff in **[802.11]**, the reduction of nonuniform access to uniform access in **[ElasticScaling]**, and the decoupling of workload and hardware in **[SoftErrors]**, resource contention and precedence constraint in **[MapReduce]**, data and resource contention in **[DatabaseSerializability]**.

3 CONCLUSION

The book provides an introduction to the techniques that are used to construct the models mentioned above. Although the computer

systems that are analyzed are complicated, the mathematical techniques often do not require more than an undergraduate education in calculus, probability and statistics.

Crafting an analytical model is therefore within reach of a performance engineer. I hope the lessons described here can encourage them to build such models for their systems, and thus help develop the science and push the envelope for engineering computer systems.

REFERENCES

[1] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing router buffers. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*. ACM, New York, NY, USA, 281–292. https://doi.org/10.1145/1015467.1015499 **[RouterBuffer]**

[2] Philip A. Bernstein, Alan Fekete, Hongfei Guo, Raghu Ramakrishnan, and Pradeep Tamma. 2006. Relaxed-currency serializability for middle-tier caching and replication. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*. ACM, New York, NY, USA, 599–610. https://doi.org/10.1145/1142473.1142540 **[DatabaseSerializability]**

[3] Diego Didona, Paolo Romano, Sebastiano Peluso, and Francesco Quaglia. 2012. Transactional auto scaler: elastic scaling of in-memory transactional data grids. In *Proceedings of the 9th International Conference on Autonomic Computing (ICAC '12)*. ACM, New York, NY, USA, 125–134. https://doi.org/10.1145/2371536.2371559 **[ElasticScaling]**

[4] Sameh Elnikety, Steven Dropsho, Emmanuel Cecchet, and Willy Zwaenepoel. 2009. Predicting replicated database scalability from standalone database profiling. In *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys '09)*. ACM, New York, NY, USA, 303–316. https://doi.org/10.1145/1519065.1519098 **[DatabaseScalability]**

[5] Bin Fan, David G. Andersen, Michael Kaminsky, and Konstantina Papagiannaki. 2010. Balancing throughput, robustness, and in-order delivery in P2P VoD. In *Proceedings of the 6th International Conference (Co-NEXT '10)*. ACM, New York, NY, USA, Article 10, 12 pages. https://doi.org/10.1145/1921168.1921182 **[P2PVoD]**

[6] Piyush Gupta and P. R. Kumar. 2006. The capacity of wireless networks. *IEEE Trans. Inf. Theor.* 46, 2 (Sept. 2006), 388–404. https://doi.org/10.1109/18.825799 **[WirelessCapacity]**

[7] Vishal Gupta and Ripal Nathuji. 2010. Analyzing performance asymmetric multicore processors for latency sensitive datacenter applications. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems (HotPower'10)*. USENIX Association, Berkeley, CA, USA, 1–8. http://dl.acm.org/citation.cfm?id=1924920.1924923 **[DatacenterAMP]**

[8] Armin Heindl, Gilles Pokam, and Ali-Reza Adl-Tabatabai. 2009. An analytic model of optimistic software transactional memory. In *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2009, April 26-28, 2009, Boston, Massachusetts, USA, Proceedings*. 153–162. https://doi.org/10.1109/ISPASS.2009.4919647 **[TransactionalMemory]**

[9] Jen-Cheng Huang, Joo Hwan Lee, Hyesoon Kim, and Hsien-Hsin S. Lee. 2014. GPUMech: GPU performance modeling technique based on interval analysis. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-47)*. IEEE Computer Society, Washington, DC, USA, 268–279. https://doi.org/10.1109/MICRO.2014.59 **[GPU]**

[10] Jongman Kim, Dongkook Park, Chrysostomos Nicopoulos, N. Vijaykrishnan, and Chita R. Das. 2005. Design and analysis of an NoC architecture from performance, reliability and energy perspective. In *Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems (ANCS '05)*. ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/1095890.1095915 **[NoC]**

[11] Donald Kossmann, Tim Kraska, and Simon Loesing. 2010. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*. ACM, New York, NY, USA, 579–590. https://doi.org/10.1145/1807167.1807231 **[CloudTransactions]**

[12] Yasuko Matsubara, Yasushi Sakurai, B. Aditya Prakash, Lei Li, and Christos Faloutsos. 2012. Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 6–14. https://doi.org/10.1145/2339530.2339537 **[InformationDiffusion]**

[13] Arun A. Nair, Stijn Eyerman, Lieven Eeckhout, and Lizy Kurian John. 2012. A first-order mechanistic model for architectural vulnerability factor. In *39th International Symposium on Computer Architecture (ISCA 2012), June 9-13, 2012, Portland, OR, USA*. 273–284. https://doi.org/10.1109/ISCA.2012.6237024 **[SoftErrors]**

[14] Angeles Navarro, Rafael Asenjo, Siham Tabik, and Calin Cascaval. 2009. Analytical modeling of pipeline parallelism. In *Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques (PACT '09)*. IEEE Computer Society, Washington, DC, USA, 281–290. https://doi.org/10.1109/PACT.2009.28 **[PipelineParallelism]**

[15] Georges Nogueira, Bruno Baynat, and Pierre Eisenmann. 2005. An analytical model for the dimensioning of a GPRS/EDGE network with a capacity constraint on a group of cells. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom '05)*. ACM, New York, NY, USA, 215–227. https://doi.org/10.1145/1080829.1080852 **[GPRS]**

[16] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. 1998. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '98)*. ACM, New York, NY, USA, 303–314. https://doi.org/10.1145/285237.285291 **[TCP]**

[17] Nilabja Roy, Abhishek Dubey, Aniruddha Gokhale, and Larry Dowdy. 2011. A capacity planning process for performance assurance of component-based distributed systems. In *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering (ICPE '11)*. ACM, New York, NY, USA, 259–270. https://doi.org/10.1145/1958746.1958784 **[PerformanceAssurance]**

[18] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. 2003. Data MULES: Modeling a three-tier architecture for sparse sensor networks. In *Proc. IEEE Workshop on Sensor Network Protocols and Applications*. 30–41. **[SensorNet]**

[19] Y. C. Tay and K. C. Chua. 2001. A capacity analysis for the IEEE 802.11 MAC protocol. *Wirel. Netw.* 7, 2 (March 2001), 159–171. https://doi.org/10.1023/A:1016637622896 **[802.11]**

[20] Y.-C. Tu, J. Sun, and S. Prabhakar. 2004. Performance analysis of a hybrid media streaming system. In *Proc. ACM/SPIE Conf. on Multimedia Computing and Networking (MMCN)*. 69–82. **[MediaStreaming]**

[21] Emanuel Vianna, Giovanni Comarela, Tatiana Pontes, Jussara M. Almeida, Virgílio A. F. Almeida, Kevin Wilkinson, Harumi A. Kuno, and Umeshwar Dayal. 2013. Analytical performance models for MapReduce workloads. *International Journal of Parallel Programming* 41, 4 (2013), 495–525. https://doi.org/10.1007/s10766-012-0227-4 **[MapReduce]**

[22] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (April 2009), 65–76. https://doi.org/10.1145/1498765.1498785 **[Roofline]**

[23] Xiaolan Zhang, Giovanni Neglia, Jim Kurose, and Don Towsley. 2007. Performance modeling of epidemic routing. *Comput. Netw.* 51, 10 (July 2007), 2867–2891. https://doi.org/10.1016/j.comnet.2006.11.028 **[EpidemicRouting]**