On Bottleneck Analysis in Stochastic Stream Processing

Raj Rao Nadakuditi, University of Michigan
Igor L. Markov, University of Michigan

Past improvements in clock frequencies have traditionally been obtained through technology scaling, but most recent technology nodes do not offer such benefits. Instead, *parallelism* has emerged as the key driver of chip-performance growth. Unfortunately, efficient simultaneous use of on-chip resources is hampered by sequential dependencies, as illustrated by Amdahl's law. Quantifying achievable parallelism in terms of provable mathematical results can help prevent futile programming efforts and guide innovation in computer architecture toward the most significant challenges. To complement Amdahl's law, we focus on stream processing and quantify performance losses due to stochastic runtimes. Using spectral theory of random matrices, we derive new analytical results and validate them by numerical simulations. These results allow us to explore unique benefits of stochasticity and show how and when they outweigh the costs for software streams.

Categories and Subject Descriptors: C.4 [**Computer Systems Organization**]: Performance of systems—*Modeling techniques*

General Terms: stream processing, stochasticity, random matrices

Additional Key Words and Phrases: stream processing, stochasticity, random matrices

1. INTRODUCTION

Parallel processing allows designers to lower clock frequencies necessary to achieve desired throughput by dividing computation between multiple computational cores. This can significantly improve power-performance trade-offs and boost chip performance beyond clock-frequency limitations. However, many applications do not exhibit straightforward parallelism and require significant restructuring, as well as support from the operating system and CPU architecture. To this end, achieving efficient parallelism through hardware engineering and improved software stack has been a key challenge in electronic system design [Asanovic et al. 2009].

Past experience with attempts at greater parallelism suggests a pattern of *diminishing returns* exemplified by Amdahl's law [Amdahl 1967].[1] Its most immediate conclusion is that a narrow focus on component improvement usually yields a smaller benefit than intuitively expected. Amdahl's law also shows that each new processor contributes less *usable power* than the previous processor. Applied to software programs with sequential dependencies, Amdahl's law helps determine where speed-ups would be most beneficial.

Amdahl's law was refined for multiple active tasks in [Agrawal et al. 2006]. In particular, the single-chip and full-system performance can be scaled significantly through *streaming* — a form of parallelism achieved by processing several dependent tasks simultaneously on unrelated data, such that job $k+1$ can commence before job

[1] Amdahl's law assumes a chain of tasks and upper-bounds the expected overall performance improvement when only one task is improved [Hill and Marty 2008].

k is finished. Streaming is particularly effective when each processing stage is implemented in dedicated hardware, e.g., graphics pipelines in modern GPUs contain up to 200 specialized processing stages. Wireless communications, cryptography, and video decoding are also processed by deep pipelines with such dedicated stages as fast Fourier transforms (FFT), discrete cosine transforms (DCT), finite-impulse response filters (FIR), Viterbi coding, standard cryptographic primitives (AES, MD5, DSA), motion estimation. Dedicated circuits offer greater performance and lower power than equivalent software implementations. They remain busy when processing multiple batches of structurally similar data — pixel patches, voice frames, encrypted blocks, etc. Similar effects can be observed in embedded systems with task-specific CPUs that support customized instructions, zero-overhead loops, and special-purpose interconnect: high-end printers, cameras and GPS navigation systems use up to ten ARM- and Tensilica-style CPUs apiece. In software, processing streams are illustrated by EDA tool-chains, where different blocks of a chip can be streamed through synthesis, placement and routing, static timing analysis (STA), design-rule checking (DRC), as well as design for manufacturing (DFM).

To limit idle time and power consumption of stream processors, stage execution times must be balanced. For example, consider a three-stage pipeline with stage execution times 1, 2 and 3. Stage 3 is always busy, while stages 1 and 2 idle two out of three cycles and one out of three cycles, respectively. Thus, instead of producing nine units of work every three cycles, the pipeline produces only six units of work, i.e., runs 33% idle. This example illustrates a design weakness, where work was not equally partitioned among the stages. However, even perfect design-time partitioning of work among stages cannot fully account for irregular or unpredictable input. For example, audio frames with a busy signal can be decoded faster than normal voice frames; some video frames exhibit less motion than others. A real-life audio-video decoder with stochastic stage-processing times is studied in [Manolache et al. 2007a]. In a different domain, the performance of GPGPU programs processing irregular data is hard to predict accurately due to (i) long graphics pipelines and (ii) increasing user-hardware separation encouraged by CUDA programming. Networks-on-chip (NoCs) experience similar challenges, but can handle more irregular compute loads — real-time NoC applications with stochastic execution times are studied in [Manolache et al. 2007b], whereas statistical-physics techniques were applied to NoC characterization in [Bogdan and Marculescu 2009] and exploited for performance optimization as in [Bogdan and Marculescu 2011a]. Stochasticity in processing rates arises from non-uniform memory access in the IBM/Sony/Toshiba Cell processor and from unpredictable cache misses in streaming software. Statistical traffic analysis in multicore platforms was studied in [Bogdan and Marculescu 2011b]. From the software perspective, randomized algorithms (such as simulated annealing, Fiduccia-Mattheyses netlist partitioning and Boolean satisfiability solvers with random restarts) exhibit stochastic runtimes and, sometimes, large-scale statistical phenomena such as phase transitions.

Our work develops performance modeling of processing pipelines (streams). We describe (stochastic) stage execution times by random variables and compare the performance of such stochastic pipelines to the performance of their deterministic variants, where execution times (latencies) correspond to the means of original

random variables. Given that stochastic variants require longer time to complete processing than deterministic variants, we quantify these losses in processing efficiency. Important research questions target the magnitude of those losses and their sensitivity to changes in pipeline configurations. Anticipating pipelines/streams with numerous stages, we study the scaling of efficiency losses with the number of stages. We analytically derive scaling trends and observe good fits to numerical simulations.

One remarkable scaling trend is observed for pipelines with a single bottleneck where all stages can simultaneously process streaming data (this setting mirrors Amdahl's analysis, but with looser constraints on parallelism). Here, we analytically derive and numerically confirm an unexpected *phase-transition*[2] — speeding up a bottleneck (by allocating greater CPU resources) brings (*i*) diminishing returns until the threshold is reached and (*ii*) no returns past the threshold, *even when the bottleneck is improved*. These trends hold for a broad range of stage-time distributions (although we start our exposition by using exponential distributions).

The phase transition separates a regime in which the presence of a finite $o(n)$ number of slow or bottleneck stages results in the latency being normally distributed with variance having an $O(n)$ leading order term to one in which the bottleneck servers are present but the latency has a Tracy-Widom distribution with having a $O(n^{2/3})$ leading order term, which also corresponds to the distribution and scaling when there are no slow servers.

In addition to the costs of stochasticity in stream processing, we note opportunities for exploiting stochasticity to improve performance. Mean stage latencies can be reduced by launching independent runs, waiting for the first run to complete, and terminating remaining runs. Our analytical results enable *a comparison of costs and benefits of stochasticity* in improving bottlenecks of software streams.

The remaining material is organized as follows. Basic concepts and terminology are reviewed in Section 2 along with relevant literature. Section 3 shows how to calculate end-to-end latency of deterministic streams and contrasts the use of *queuing theory* and *random-matrix theory* in the analysis of stochastic streams. Sections 5 and 6 derive the cost of stochasticity for balanced and unbalanced streams, resp. The assumption of exponential distributions made to derive key results is overcome in Section 7. In Section 8, we quantify the benefits of stochasticity for software streams and compare them to the costs. Conclusions are given in Section 9.

[2] The term *phase transition* originally arose in thermodynamics to represents a qualitative change in statistical properties of a multi-particle system, such as freezing and evaporation of liquids, melting and sublimation of solids, condensation and deposition of gases, as well as transitions between gases and plasma. Phase transitions have been provably demonstrated in combinatorial optimization, as illustrated by the easy-hard transition of random 3-SAT instances near the 4.2 clause-to-variable ratio (confirmed empirically by the runtimes of several types of SAT solvers). Being a statistical phenomenon, phase transitions can be expected only in systems consisting of a sufficiently large number of components. This has prevented, so far, the demonstration of phase transitions in computing hardware though there have been properly identified phase transitions in MPSoC / NoC research [Ogras and Marculescu 2005; Bogdan and Marculescu 2009]. However, our work shows that future computing systems of sufficient size should exhibit phase transitions.

2. END-TO-END LATENCY ANALYSIS

In this section we outline the problem formulation addressed in our work, define necessary terminology and review prior work.

2.1 The model

Given a stream with m simultaneously active stages shown in Figure 1, we evaluate its performance on a batch of n independent jobs. Each job starts at the first stage and advances sequentially through the remaining stages — once job j has been processed by stage i, it is queued up for stage $i+1$ and processed once job $j-1$ clears that stage (Section 3 formalizes these conditions). Inter-stage FIFOs are assumed to always be sufficiently large, and all executions occur in-order. Our key performance metric is *end-to-end latency* (**EEL**) $l(m,n)$ — the completion time of the last (n-th) job at the last (m-th) stage. Figure 1 illustrates a three-stage stream and the emergence of idle periods on some stages between jobs. Unlike in [Davare et al.], (i) no end-to-end latency deadlines are imposed and, (ii) our FIFO inter-stage queuing model does not provision for explicit communication, simplifying the computation of the end-to-end latency. This assumption ensures that the stage times are job independent and hence statistically independent. For

Stochastic stage completion times arise in several contexts, including (i) extreme sensitivity of runtime to the complexity of input data, (ii) non-determinism due to randomized algorithms, shared resources, interrupts, and cache misses, as well as (iii) the lack of accurate information about (possibly deterministic) stage completion times. For analytic purposes, these diverse circumstances are captured by modeling stage completion times with random variables, which also makes EEL a random variable.

The main objective in this work is *to quantify the impact of the probability dis-*

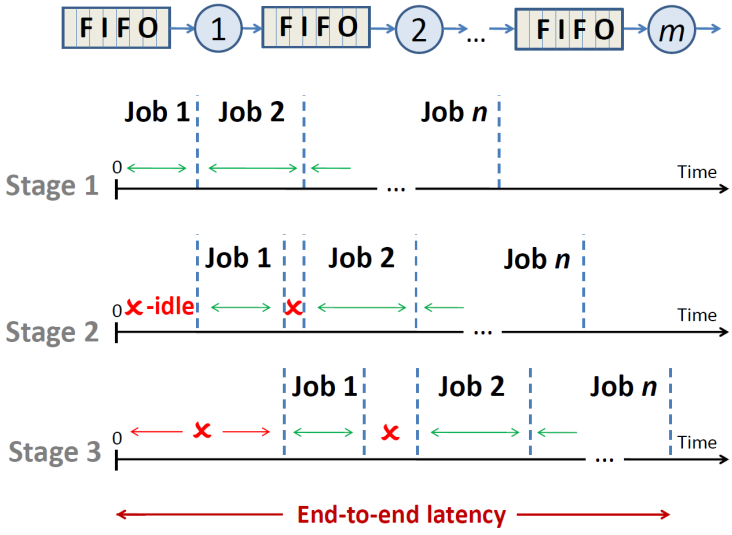

Fig. 1. A timing diagram of a stream processor. Idle periods are indicated with red crosses.

tributions of individual stage times on the end-to-end latency statistic. We seek to characterize the mean end-to-end latency (**MEEL**), the variance, and whenever possible provide a complete analytical description of EEL via its probability distribution.

Closest related work by Rajsbaum and Sidi [1994] and, more recently, by Lipman and Stout [2006], studied the impact of random processing times and transmission delays on the average number of computational steps executed by a processor in the network per unit time when attempting to synchronize over a distributed network. Our work differs in two notable ways. First, instead of bounds, we directly describe relevant probability distributions. Second, while we start off our exposition in terms of exponential probability distributions, we later conclude that the specific form of the probability distributions matters less than anticipated. In particular, the new scaling phenomena we discover for the EEL statistic hold for a broad class of stage-time probability distributions.

Performance bottlenecks are of particular interest in our work, for the same reasons as they are in Amdahl's law. However, in the context of stream processing with balanced stages and *stochastic stage times*, the time distribution of a bottleneck stage may exhibit a *greater variance* or *longer tail*. This observation motivates designers to collect runtime statistics as in [Chrysos et al. 1998] so that such bottleneck stages can be identified and their impact mitigated, e.g., by allocating additional compute resources.[3] In practice, each stage may exhibit a different

[3]If stage times are independent, then processing the same data at the same stage on multiple processors (and using the first available result) can reduce variance and shorten the tail of the resulting time distribution.

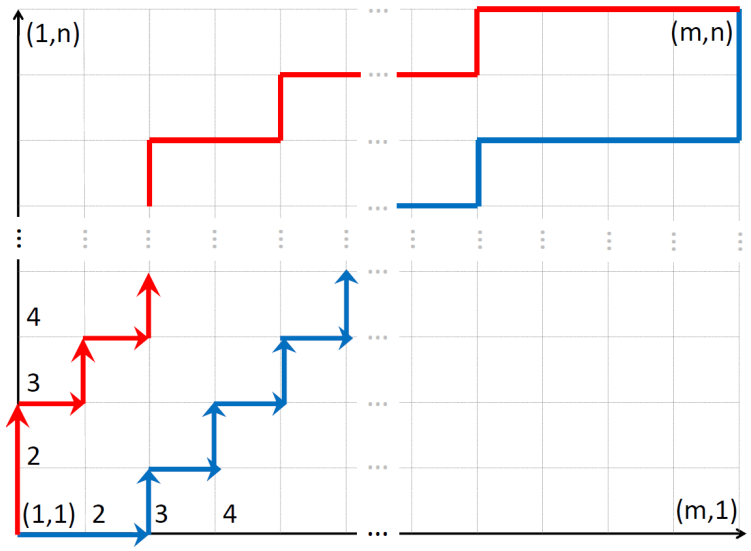

Fig. 2. Monotonic paths $(1,1) \to (m,n)$ used to model end-to-end latency from Figure 1 by Formula 3 with $k = m$.

runtime distribution, whereas hardware designers, compiler experts and software developers have no simple way to locate bottlenecks. Even with existing profiling tools, pinpointing the "features" of runtime distribution (large variance, long tail) that affect end-to-end latency most remains difficult. Indeed, bottleneck identification and mitigation in stochastic streams have so far been more art than science. Design trade-offs to satisfy power constraints and resource limitations have been performed by trial and error.

3. MATHEMATICAL BACKGROUND

This section first reviews latency computation of deterministic queues. For queues with stochastic service times, we outline two types of techniques to compute latency distributions and compare them.

Notation. We employ the following notation throughout this paper:

—S_i: Stage $i \in \{1, \ldots, m\}$ where stages labeled from 'left to right',

—C_j: Job $j \in \{1, \ldots, n\}$ where jobs are labeled from 'right to left', *i.e.*, in the order that jobs exit the system:

—$l(i, j)$: Time at which job j exits stage S_i,

—$w(i, j)$: Service time for job j at stage i.

Fundamental EEL recursion. Recall that $l(i, j-1)$ is the time at which job $j-1$ exits stage i while $l(i-1, j)$ is the time at which job j exits stage $i-1$. When $l(i, j-1) \leq l(i-1, j)$, then job j can be served immediately by stage i as soon as job j exits stage $i-1$. *In-order execution* assumed (see Section 2) implies that when $l(i, j-1) > l(i-1, j)$ then job j has to wait in server i's queue for job $j-1$ to be processed and exit stage i's queue. Hence we have the recursion:

$$l(i,j) = w(i,j) + \begin{cases} l(i-1,j) & \text{when } l(i,j-1) \leq l(i-1,j), \\ l(i,j-1) & \text{when } l(i,j-1) > l(i-1,j). \end{cases} \tag{1}$$

or equivalently,

$$\boxed{l(i,j) = \max\{l(i-1,j), l(i,j-1)\} + w(i,j)} \tag{2}$$

for all $i, j \in \mathbb{Z}$. The constraints in (2) and (3) suggest an $O(mj)$-time dynamic programming algorithm for computing $l(m, j)$. The fundamental recursion derived above can be recast as the directed *last passage percolation* (LPP) problem:

$$l(i,j) = \max_{\pi \in P(i,j)} \left(\sum_{(k,\ell) \in \pi} w(k,\ell) \right), \tag{3}$$

where $P(i, j)$ is the set of 'up/right paths' ending at (i, j), *i.e.*, $\pi \in P(i, j)$ if $\pi = \{(k_s, \ell_s)\}_{s=-\infty}^{0}$ such that $(k_0, \ell_0) = (i, j)$ and $(k_s, \ell_s) - (k_{s-1}, \ell_{s-1})$ is either $(1, 0)$ or $(0, 1)$ for all $s \leq 0$. Note that the right-hand-side of (3) satisfies the same recurrence as $l(i, j)$ in (2) since a path in $P(i, j)$, as shown in Figure 2, consists of either a path in $P(i-1, j)$ and (i, j), or a path in $P(i, j-1)$ and (i, j). These monotonic paths capture all possible critical paths during stream's execution. We give *closed-form expressions* for $l(k, n)$ for two cases in discussions after Formulas 11 and 21, and contrast them with results for the stochastic case.

This recursion can be found in the seminal paper by Glynn and Whitt [1991] wherein the authors attribute the formulation to Tembe and Wolff [1974].

Stochastic queuing theory Glynn and Whitt [1991] studied the statistics of Formula 3 when $m \gg n$ and *vice versa*. For stream processing, this assumption can be justified in the traditional setting where the number of stream stages remains limited, i.e., $m = O(1)$, but the number of jobs is large. Under these assumptions EEL is *normally* distributed via the law of large numbers [Glynn and Whitt 1991]. Consequently, the asymptotic scaling of the mean is straightforward, and the impact of a small number of bottlenecks is what one would intuitively expect. We note that the "interacting-particle system" interpretation [Srinivasan 1993] used by queuing theory simplifies the analysis by neglecting the interaction between stages — this is a reasonable assumption when $m \gg n$ or $n \gg m$, but not when n and m are both small or when both are large.

Numerous parallel cores can be useful in deep streams when the number of streaming jobs is sufficiently high. To this end, the RAMP project at Berkeley is developing a massive FPGA-based emulator to study large-scale behavior of many-core systems, recently reaching the 1008-processor milestone [Burke et al. 2008]. However, current supercomputers integrate 300,000 cores, and "supercomputers with 100 million cores are coming by 2018" [Thibodeau 2009]. This motivates our focus on *analytical estimates*. When both n and m are large in the stream model of Section 2, the interactions between stochastic stage-time distributions accumulate, and the assumptions made in queuing theory are no longer valid (see discussion after Formula 11). The Gaussian distribution predicted by queuing theory transitions into the *type-2 Tracy-Widom* distribution studied in the *spectral theory of random matrices* [Johansson 2000; Johnstone 2001], and the asymptotic scaling of variance changes as well. Figure 3 contrasts the two distributions.

The type-2 Tracy-Widom distribution (TW_2) describes the largest eigenvalue of random Hermitian matrices [El Karoui 2007] and arises in combinatorics. If π is a random n-element permutation, then the length of the *longest increasing subsequence* of π converges (with appropriate scaling and recentering) to the TW_2 distribution as $n \to \infty$ [Baik et al. 1999]. For exponentially-distributed stage times, Formula 3 is related to the LONGEST INCREASING SUBSEQUENCE PROBLEM. Empirical evidence in [Deift 2007] suggests viewing the TW_2 distribution as a nonlinear variant of the law of large numbers for EEL. Thus, we use TW_2 and related mathematics to perform accurate analysis of stochastic streams.

4. PERTINENT RESULTS FROM RANDOM MATRIX THEORY

In general, it is not easy to find an explicit formula of cumulative distribution function (c.d.f.) of $l(i, j)$. However, during the last ten years or so, a development in the *last-passage percolation* problem (LPP) established a computationally easy formula of c.d.f. for a special choice of the waiting times which we heavily exploit for the analytical results presented in this paper. [4] We first restate a textbook

[4]Recent work on statistical traffic analysis in NoCs and multicore platforms [Bogdan and Marculescu 2009; 2011b] stresses the importance of going beyond exponential distributions studied in queuing theory. We agree with this assessment and note that while our results were derived assuming exponentials, they appear to carry over in a more general context.

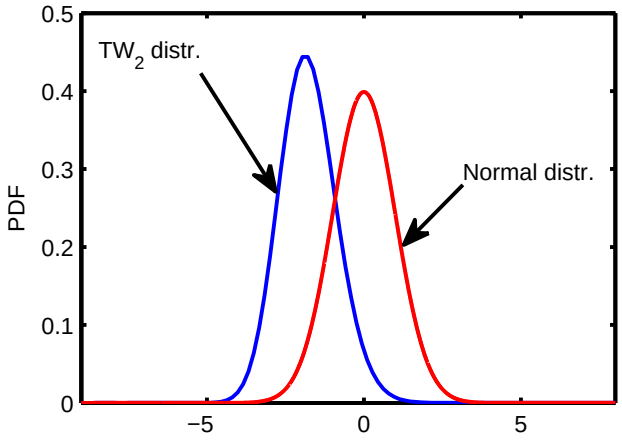

Fig. 3. The Tracy-Widom and normal distributions.

definition to establish basic notation.

DEFINITION 4.1 EXPONENTIAL DISTRIBUTION. *Throughout this paper, for* $\lambda > 0$*, we denote by* $f(t;\lambda)$ *the exponential distribution with the p.d.f.*

$$f(t;\lambda) = (1/\lambda)\exp(-t/\lambda), \qquad t \geq 0. \tag{4}$$

For this choice of normalization, the mean $E[t] = \lambda$ *and variance* $var(t) = \lambda^2$*.*

DEFINITION 4.2 SOLVABLE LPP MODEL [BORODIN AND PÉCHÉ 2008]. *Consider real-valued numbers* $a_i, b_j \in (-\infty, \infty]$ *with* $a_i + b_j > 0$ *for all* $i, j \in \mathbb{Z}$*. Let* $w(i,j)$ *be an exponentially distributed random variable, normalized as in Definition 4.1 so that it has mean* $a_i + b_j$*. Let the* $w(i,j)$*'s for* $i, j \in \mathbb{Z}$*, be independent. Consider the LPP problem:*

$$l(m,n) = \max_{\pi \in P} \left(\sum_{(k,\ell)\in\pi} w(k,\ell) \right) \tag{5}$$

where P *is the set of all up/right paths starting from* $(0,0)$ *and ending at* (m,n) *with* $w(i,j)$ *being random numbers chosen as above.*

The theorem stated next, explicitly connects the distribution of $l(m,n)$ with $w(i,j)$'s chosen as in Definition 4.2 with the largest eigenvalue $\lambda_{\max}(\cdot)$ of a specific random matrix.

PROPOSITION 4.1 [BORODIN AND PÉCHÉ 2008]. *Let* X *be an* $m \times n$ *matrix with independent entries whose distributions are given by*

$$X_{ij} \sim \mathcal{CN}(0, a_i + b_j), \tag{6}$$

where $\mathcal{CN}$ *denotes the circularly symmetric, complex normal distribution. Then we have that*

$$\lambda_{\max}(XX^*) \stackrel{\mathcal{D}}{=} l(m,n), \tag{7}$$

where $\stackrel{\mathcal{D}}{=}$ *denotes equality in distribution.*

For proof, see [Borodin and Péché 2008, Proposition 1].

5. ANALYSIS OF BALANCED STOCHASTIC STREAMS

The research strategy pursued in this work is to initiate analysis in terms of balanced *exponentially* distributed stage times with mean λ and standard deviation λ.

The initial choice of exponential distributions is motivated by its special role as a *worst case* distribution from an information-theoretic perspective, as described next. Extensions of the key results to a broader class of probability distributions are discussed in Section 7. In a practical setting, we might not know the entire stage-time distribution, but we can usually estimate its mean. From the many probability distributions with a given mean, we distinguish the unique distribution that maximizes the Shannon entropy[5] because it offers *the most random* probabilistic model subject to what is known. Among all probability distributions supported on $t \geq 0$ with mean λ, the *exponential distribution* exhibits maximum entropy [Cover and Thomas 2006, Chapter 11]. This *worst-case* information-theoretic argument was previously used by Rajsbaum and Sidi [1994] to motivate the focus on exponential distributions in a setting related to ours.

To leverage results from the random-matrix theory, we first form a complex-valued $m \times n$ matrix X with identically independently distributed entries, and then consider the maximal eigenvalue of XX^*.

THEOREM 5.1. *Let the m stages have service times that are exponentially distributed means $\lambda_1 = \ldots, \lambda_m =: \lambda$. Let X be a $m \times n$ random matrix with i.i.d. entries distributed as $X_{ij} \sim \mathcal{CN}(0, \lambda)$. Then for every m and n we have that*

$$l(m, n) \stackrel{\mathcal{D}}{=} \lambda_{\max}(XX^*). \tag{8}$$

PROOF. Recall that $w(i, j)$ is the service time for job j at stage i. The assumptions imply that $w(i, j)$ for all i, j are identically distributed. Setting $a_i = \lambda$ for all

[5]A single number that is commonly used to measure the amount of uncertainty contained in a probability distribution [Cover and Thomas 2006].

		MEAN		VARIANCE	
m	n	Experiment	Theory	Experiment	Theory
5	5	13.1024	12.3685	9.4351	15.0981
10	10	30.9954	30.3849	18.6033	23.9668
20	20	68.3172	67.8858	33.0268	38.0449
40	40	145.0274	144.7371	55.1251	60.3926
80	80	300.9902	300.7699	90.0644	95.8673
160	160	615.9515	615.7717	148.8302	152.1799
320	320	**1249**.4124	**1249**.4742	236.0294	241.5705
480	480	**1885**.7545	**1885**.0567	311.7331	316.5469
640	640	**2521**.6221	**2521**.5399	374.6064	383.4693
1000	1000	**3955**.4348	**3955**.3710	506.5496	516.3498

Table I. Empirical mean and variance of end-to-end latency, computed over 1000 Monte-Carlo trials, compared to theoretical predictions in Formulas 10 and 11, respectively.

$i = 1, \ldots m$ and $b_j = 0$ for $j = \ldots 1, n$ and applying Proposition 4.1 gives us the desired result. Johansson [2000] has an alternate derivation of this result. □

5.1 The cost of stochasticity

THEOREM 5.2. *As $m, n \longrightarrow \infty$, we have that*

$$\frac{l(m,n) - \mu_{m,n}}{\sigma_{m,n}} \xrightarrow{\mathcal{D}} \mathsf{TW}_2, \tag{9}$$

where

$$\mu_{m,n} = \lambda(\sqrt{n} + \sqrt{m})^2 \text{ and } \sigma_{m,n} = \lambda \frac{(\sqrt{m} + \sqrt{n})^{4/3}}{(mn)^{1/6}},$$

and $\xrightarrow{\mathcal{D}}$ denotes convergence in distribution.

PROOF. Johnstone studied the distribution of the largest eigenvalue of the matrix XX^* where X is a complex-valued matrix $m \times n$ matrix with i.i.d. normally distributed entries with mean zero and unit variance in [Johnstone 2001]. The results follow by invoking the correspondence, established in Theorem 5.1, between the distribution of $\lambda_{\max}(XX^*)$ (given by Theorem 1.3 of [Johnstone 2001]) and $l(m,n)$. □

Since the TW_2 distribution asymptotically describes $\lambda_{\max}$ [Johnstone 2001], we are able to highlight the important qualitative trends of $l(m,n)$.

COROLLARY 5.1.

$$E\left[l(m,n)\right] = \lambda(\sqrt{n} + \sqrt{m})^2 - 1.7711\,\lambda \frac{(\sqrt{m} + \sqrt{n})^{4/3}}{(mn)^{1/6}} + o((mn)^{1/6}) \tag{10}$$

$$Var\left[l(m,n)\right] = 0.8132\,\lambda^2 \left(\frac{(\sqrt{m} + \sqrt{n})^{4/3}}{(mn)^{1/6}}\right)^2 + o((mn)^{1/6}) \tag{11}$$

PROOF. Theorem 5.2 implies that

$$l(m,n) = \mu_{m,n} + \sigma_{m,n}\,\mathsf{TW}_2 + o((mn)^{1/6}).$$

The TW_2 distribution (see Figure 3) has mean -1.7711 and variance 0.8132 [Johnstone 2001](see [Bornemann 2010] for machine-precision level accurate expressions for the mean and variance). The stated result follows by making the appropriate substitutions. □

Figure 4 illustrates scaling behavior predicted by Formula 10. In contrast, note that n identical jobs streamed through m stages with identical *deterministic* latencies λ take $\lambda(m+n)$ time. But MEEL in the stochastic case scales[6] as $\lambda(\sqrt{m} + \sqrt{n})^2 = \lambda(m + n + 2\sqrt{nm})$.

Hence, the cost of stochasticity scales as $2\lambda\sqrt{mn}$.

[6]The second term on the right hand side of Formula 10 is $O((mn)^{1/6})$ and hence can be ignored relative to the $O((mn)^{1/2})$ terms that emerge in the expansion of the first term in Formula 10.

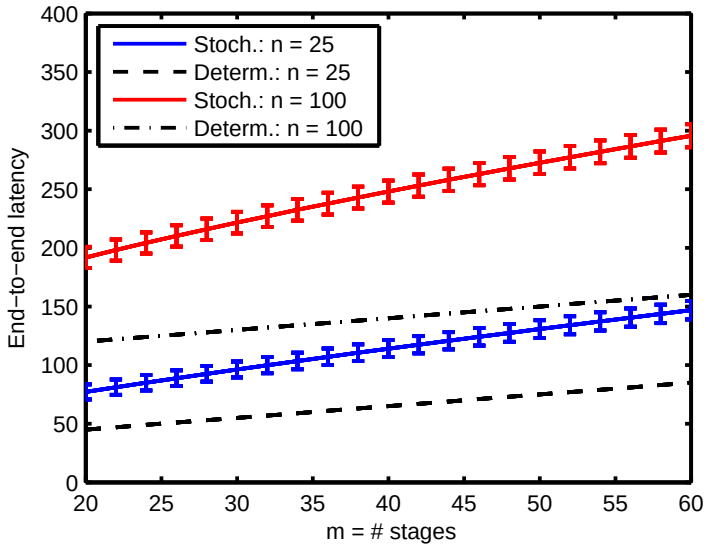

Fig. 4. Theoretical scaling of mean end-to-end latency with the number of stages for exponentially distributed stage times. Solid lines illustrate Formula 10, and error bars give standard deviation according to Equation 11. For comparison, dashed lines show latencies in a deterministic stream.

Observe that for $n \gg m$ or $m \gg n$, the term $2\lambda\sqrt{mn}$ is asymptotically negligible because $2\sqrt{mn} = o(mn)$, but it may contribute up to 50% of EEL when $m = \Theta(n)$. This first-order result is alluded to in the seminal paper on queuing theory by Glynn and Whitt [1991]. However, the *law of vanishing returns* stated next is new and exploits results from random-matrix theory [Baik et al. 2005].

5.2 A law of vanishing returns for bottleneck optimization

THEOREM 5.3. *For integer $k > 0$, let the service time of the first k stages be exponentially distributed with means $\lambda_1, \ldots, \lambda_k$ and the service time of the remaining $m - k$ stages also be exponentially distributed with means $\lambda_{k+1} = \ldots, \lambda_m =: \lambda$. Let $\lambda_i > \lambda$ for all $i = 1, \ldots, k$ so that we have k bottleneck (or slow) stages in the pipeline. Then*

$$l(m,n) \stackrel{\mathcal{D}}{=} \lambda_{\max}(\Lambda^{1/2} X X^* \Lambda^{1/2}) \tag{12}$$

where $\Lambda = diag(\lambda_1, \ldots, \lambda_k, \lambda, \ldots, \lambda)$ is an $m \times m$ diagonal matrix and X is an $m \times n$ complex-valued matrix with i.i.d. normal entries with zero mean and unit variance.

PROOF. As before $w(i,j)$ is the service time for job j at stage i. The assumptions imply that $w(i,j)$ for the non-bottleneck stages $i = k+1, \ldots, m$ and jobs $j = 1, \ldots n$ are identically distributed with exponential distribution having mean λ. Consider a bottleneck stage $i = 1, \ldots, k$; it has i.i.d. service time $w(i,j)$ for each job that is exponentially distributed with mean λ_i for all j. Setting $a_i = \lambda_i$ for $i = 1, \ldots, k$ and $a_i = \lambda$ for all $i = k+1, \ldots m$, and $b_j = 0$ for $j = \ldots 1, n$ and applying Proposition

4.1 gives us the equivalence:

$$l(m,n) \stackrel{\mathcal{D}}{=} \lambda_{\max}(YY^*).$$

Here Y is an $m \times n$ complex-valued matrix with Y_{ij} having a variance $a_i = \lambda_i$ for $i = 1, \ldots, k$ and $a_i = \lambda$ for $i = k+1, \ldots, m$. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k, \lambda, \ldots, \lambda)$ be an $m \times m$ diagonal matrix. Then, simple matrix calculus and the properties of Gaussian random variables shows that $Y \stackrel{\mathcal{D}}{=} \Lambda^{1/2} X$ where X is an $m \times n$ complex-valued matrix with i.i.d. zero mean, unit variance normally distributed entries. The $YY^* \stackrel{\mathcal{D}}{=} \Lambda^{1/2} X X^* \Lambda^{1/2}$ and the result follows. □

REMARK 5.1. *Note that in the hypothesis of Theorem 5.3 we set the first k stages to represent the bottleneck stages for the sake of expositional brevity. Any k of the m stages could be bottleneck stages and the results would still apply.*

THEOREM 5.4. *Define* $1/c = \lambda(1 + \sqrt{m/n})$.
a) When $\max(\lambda_1, \ldots, \lambda_k) < 1/c$ *we have that as* $m, n \longrightarrow \infty$,

$$\frac{l(m,n) - \mu_{m,n}}{\sigma_{m,n}} \xrightarrow{D} \mathsf{TW}_2, \tag{13}$$

where

$$\mu_{m,n} = \lambda(\sqrt{n} + \sqrt{m})^2 \text{ and } \sigma_{m,n} = \lambda \frac{(\sqrt{m} + \sqrt{n})^{4/3}}{(mn)^{1/6}},$$

as in Theorem 5.2.

b) When $\lambda_{\max} := \max(\lambda_1, \ldots, \lambda_k) > 1/c$ *(and assuming that there is only one index i for which* $\lambda_i = \lambda_{\max}$*), we have that as* $m, n \longrightarrow \infty$,

$$\frac{l(m,n) - \mu_{m,n}}{\sigma_{m,n}} \xrightarrow{D} \mathcal{N}(0,1), \tag{14}$$

where

$$\mu_{m,n} = \lambda_{\max}\left(n + \frac{m}{\lambda_{\max} - \lambda}\right) \text{ and } \sigma^2_{m,n} = \lambda^2_{\max}\left(n - \frac{m}{(\lambda_{\max} - \lambda)^2}\right).$$

PROOF. This result was established in [Baik et al. 2005]. For an alternate, matrix-theoretic derivation that directly exploits the connection between the largest eigenvalue of a random matrix and $l(m,n)$ established in Theorem 5.3 see [Nadakuditi and Silverstein 2010; Nadakuditi and Benaych-Georges 2010]. □

COROLLARY 5.2. *a) When* $\lambda_{\max} := \max(\lambda_1, \ldots, \lambda_m) < 1/c$*, we have that*

$$E\left[l(m,n)\right] = \mu_{m,n} - 1.7711\,\sigma_{m,n} + o((mn)^{1/6}) \tag{15}$$

$$Var[l(m,n)] = 0.8132\,\sigma^2_{m,n} + o((mn)^{1/6}). \tag{16}$$

Here $\mu_{m,n}$ *and* $\sigma_{m,n}$ *are given by Theorem 6.2-a).*

b) When $\lambda_{\max} := \max(\lambda_1, \ldots, \lambda_m) > 1/c$*, we have that*

$$E\left[l(m,n)\right] = \mu_{m,n} + o((mn)^{1/4}) \tag{17}$$

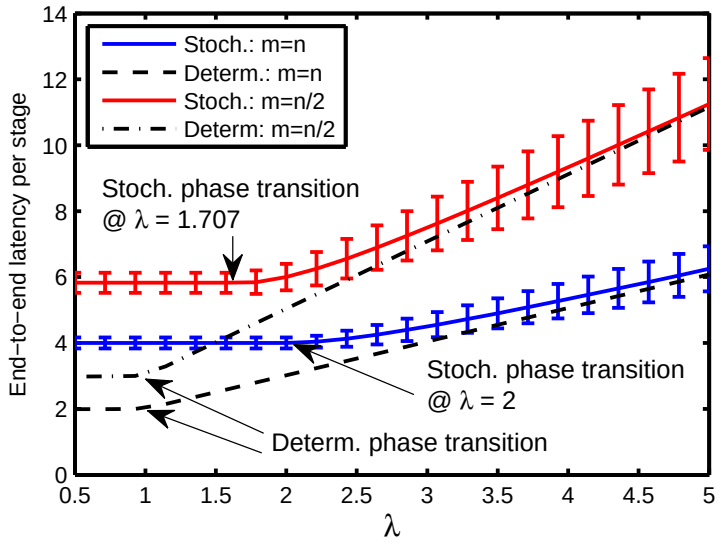

Fig. 5. The effect of a *single bottleneck* stage with $\lambda := \lambda_{\max} > 0$ when the other stages have $\lambda_2 = \ldots, \lambda_m = 1$. End-to-end latency (normalized per stage) given by Corollary 5.2 exhibits a *phase transition* at the critical value $\tau = 1 + \sqrt{\frac{m}{n}}$. Error bars show standard deviation as per Formula 11. Dashed lines give a deterministic baseline as in Figure 4.

$$Var[l(m,n)] = \sigma^2_{m,n} + o((mn)^{1/4}). \tag{18}$$

Here $\mu_{m,n}$ and $\sigma_{m,n}$ are given by Theorem 6.2-b).

PROOF. The result in part a) follows by employing the argument used in the proof of Corollary 5.1. The result in part b) follows from the same argument applied to the normal distribution with mean zero and variance one. □

Figure 5 plots the normalized mean and variance *per stage* to illustrate the emergent scaling behavior for the setting where $\lambda_2 = \ldots = \lambda_m = 1$: when the mean of the bottleneck-stage time is below the critical threshold $\tau = 1/c = (1 + \sqrt{m/n})$, then, surprisingly, the end-to-end latency of the system becomes insensitive to changes in λ_1. The same holds for $o(n)$ bottlenecks. This result can be interpreted as an analog of Amdahl's law, for stream processing with stochastic runtime distributions.

Numerical validation of the formulas presented so far was performed by extensive Monte-Carlo simulations in MATLAB. Table I shows excellent agreement between analytical results and numerical simulations. Figure 6 graphically illustrates empirical accuracy of our bottleneck predictions. Notice that the errors decrease as parameters grow — this is expected for asymptotic estimates. The variances in Table I appear to be *over-estimated* in our empirical results, betraying (distribution-dependent) higher-order terms missing from our estimates.

The phase transition can be intuitively interpreted as described next. Consider the setting where there are $m-1$ stages having the same mean service time so that $\lambda_1, \ldots, \lambda_{m-1} = 1$ followed by a single bottleneck server with mean service time

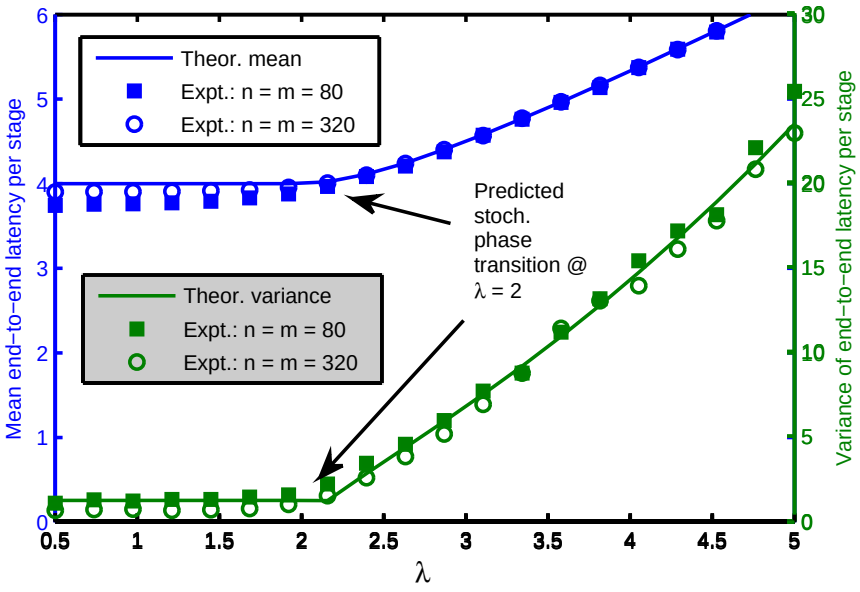

Fig. 6. Empirical evaluation of analytical predictions (solid lines) when $\lambda_2 = \ldots = \lambda_m = 1$ in Corollary 5.2 for the mean (left axis) and the variance (right axis) against $\lambda := \lambda_{\max} > 0$ of a single bottleneck stage. Datapoints are averaged over 1000 Monte-Carlo trials.

$\lambda_m \geq 1$. Let us examine the quantity $l(m,n)$ by conditioning on the number of jobs xm, where $x \in [0,1]$, waiting to be serviced when job n enters to queue for stage m. Then we have that (to leading order)

$$E[l(m,n)|x] \approx xn\lambda_m + (\sqrt{(1-x)n} + \sqrt{m})^2,$$

where the first term is nx times the average service time of the bottleneck stage while the second term, by Corollary 5.1, is the amount of time it would have taken for the $n - nx$ jobs to be processed by all m stages. Let $f(x) = E[l(m,n)]/n$ so that for $x \in [0,1]$, we have

$$f(x;\lambda_m) = \frac{xn\lambda_m + (\sqrt{(1-x)n} + \sqrt{m})^2}{n} = \lambda_m x + (\sqrt{1-x} + \sqrt{\frac{m}{n}})^2. \qquad (19)$$

Equation (19) captures the dependence of the EEL of job n as a function of the proportion of jobs that are 'held up' by the bottleneck stage. By this viewpoint, the quantity

$$\max_{x\in[0,1]} f(x),$$

captures the maximum latency incurred due to all possible proportions of jobs held

up by the bottleneck server. A simple calculation shows that

$$\max_{x\in[0,1]} f(x) = \begin{cases} f(0) = \left(1+\sqrt{\frac{m}{n}}\right)^2 & \text{if } \lambda_m < 1+\sqrt{\frac{m}{n}} \\ \lambda_m\left(1+\frac{m/n}{\lambda_m-1}\right) & \text{otherwise.} \end{cases} \tag{20}$$

This yields the location of the phase transition, denoted by $1/c$, in Corollary 5.2. Note that whether the first stage or the last stage (or an in-between stage) is the bottleneck does not change the answer - we set the bottleneck stage to be the last stage for expositional simplicity. It is only when $\lambda_m > 1+\sqrt{m/n} = 1/c$, that the EEL is dominated by the bottleneck server, in which case, by the law of large numbers the EEL becomes normally distributed.

6. ANALYSIS OF UNBALANCED STOCHASTIC STREAMS

We now generalize the previous setting by assuming that the m stage times are independent and exponentially distributed with *different* parameters $\lambda_1, \ldots, \lambda_m$. In Section 7, we discuss how these results provide insight for the setting where the streams have balanced means but unbalanced variances.

THEOREM 6.1. *For integer $k > 0$, let the m stages have service times that are exponentially distributed with means $\lambda_1, \ldots, \lambda_m$. Then*

$$l(m,n) \stackrel{\mathcal{D}}{=} \lambda_{\max}(\Lambda^{1/2} X X^* \Lambda^{1/2}),$$

where $\Lambda = diag(\lambda_1, \ldots, \lambda_m)$ is an $m \times m$ diagonal matrix and X is an $m \times n$ complex-valued matrix with i.i.d. normal entries with zero mean and unit variance.

PROOF. We employ the same approach as in the proof of Theorem 5.3. Setting $k = m$ gives us the stated result. □

THEOREM 6.2. *Let c be the unique solution in $[0, 1/\max(\lambda_1, \ldots, \lambda_m)]$ of the equation:*

$$\sum_{i=1}^{m} \left(\frac{\lambda_i c}{1-\lambda_i c}\right)^2 = n. \tag{21}$$

a) When $\max(\lambda_1, \ldots, \lambda_m) < 1/c$ we have that as $m, n \longrightarrow \infty$,

$$\frac{l(m,n) - \mu_{m,n}}{\sigma_{m,n}} \xrightarrow{D} \mathsf{TW}_2, \tag{22}$$

where

$$\mu_{m,n} = \frac{1}{c}\left(n + \sum_{i=1}^{m} \frac{\lambda_i c}{1-\lambda_i c}\right) \tag{23}$$

and

$$\sigma^3_{m,n} = \frac{n}{c^3}\left(1 + \frac{1}{n}\sum_{i=1}^{m}\left(\frac{\lambda_i c}{1-\lambda_i c}\right)^3\right). \tag{24}$$

b) When $\lambda_{\max} := \max(\lambda_1, \ldots, \lambda_m) > 1/c$ *(and assuming that there is only one index* i *for which* $\lambda_i = \lambda_{\max}$*), we have that as* $m, n \longrightarrow \infty$

$$\frac{l(m,n) - \mu_{m,n}}{\sigma_{m,n}} \xrightarrow{D} \mathcal{N}(0,1), \tag{25}$$

where

$$\mu_{m,n} = \lambda_{\max}\left(n + \sum_{i=1}^{m} \frac{\lambda_i}{\lambda_i - \lambda_{\max}}\right) \tag{26}$$

and

$$\sigma^2_{m,n} = \lambda^2_{\max}\left(n - \sum_{i=1}^{m}\left(\frac{\lambda_i}{\lambda_i - \lambda_{\max}}\right)^2\right). \tag{27}$$

Proof. El Karoui [2007] studied the distribution of the largest eigenvalue of the matrix $\Lambda^{1/2}XX^*\Lambda^{1/2}$ where X is a complex-valued matrix $m \times n$ matrix with i.i.d. normally distributed entries with mean zero and unit variance and Λ is an arbitrary diagonal matrix. The results follow by invoking the correspondence, established in Theorem 6.1, between the distribution of $\lambda_{\max}(\Lambda^{1/2}XX^*\Lambda^{1/2})$ and $l(m,n)$. Specifically, part a) appears in Theorem 1 of [El Karoui 2007] and part b) in [Baik et al. 2005; El Karoui 2007]. The location of the phase transition is also derived in [Nadakuditi and Silverstein 2010]. □

Theorem 6.2 reveals the existence of a phase transition in the EEL distribution depending on how distinct the mean service times at stage are. If they are closely clustered and below the critical threshold c, then the TW_2 arises; otherwise we get the normal distribution. Consequently, the MEEL experience a phase transition for the mean and variance of the EEL as well, exactly as stated in Corollary 5.2 with c now given by the solution of Equation 21. Note that, accordingly, we substitute $\mu_{m,n}$ and $\sigma_{m,n}$ given by Theorem 6.2-a) and b) for the mean and variance computation in Corollary 5.2.

As before, we can interpret the phase transition location as the λ value at which

$$\max_{x\in[0,1]} f(x) := \lambda_m x + l(m, n - nx)$$

The value for λ_m at which $\lambda_m x = l(m, n - nx)$ corresponds precisely to the critical value denoted by $1/c$ in Theorem 6.2 note that the symmetries in the problem imply that whether the first stage or the last stage (or an in-between stage) is the bottleneck does not change the answer. It is only when $\lambda_m > 1/c$, that the EEL is dominated by the bottleneck stage, whence, by the law of large numbers, the EEL becomes normally distributed.

Note that n identical jobs streamed through m stages with deterministic latencies $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ take $\sum_{i=1}^{m} \lambda_i + n\lambda_1$ time. But in the stochastic case, MEEL scales with $\mu_{n,m}$ as given by Formula 23, and the *cost of stochasticity* scales as

$$\frac{1}{c}\left(n + \sum_{i=1}^{m} \frac{\lambda_i c}{1 - \lambda_i c}\right) - \left(\sum_{i=1}^{m} \lambda_i + n\lambda_1\right) > 0 \tag{28}$$

where c is the solution of Equation 21.

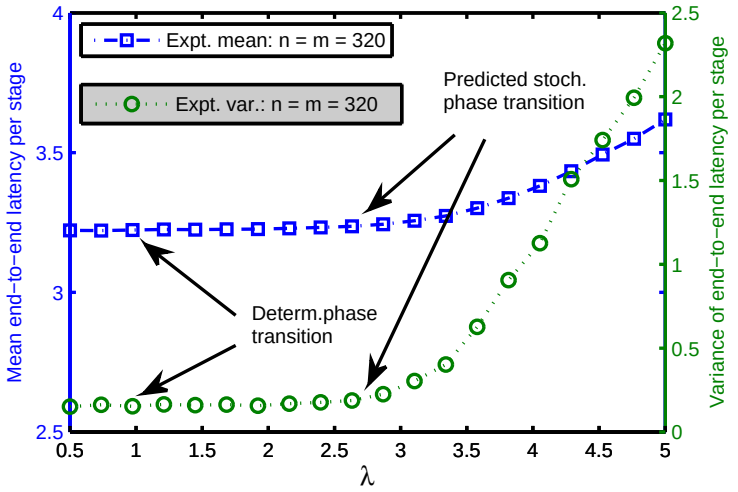

Fig. 7. The law of vanishing returns for an unbalanced stochastic stream with normally-distributed stage-times with a single bottleneck stage having mean stage time λ. Empirical datapoints are overlaid against theoretical predictions (lines) for the mean (left axis) and the variance (right axis) of end-to-end latency.

The realization that the costs of stochasticity can be significant leads us to optimization. Since such optimizations typically focus on bottleneck stages, it is useful to characterize those stages. Suppose that $m-1$ stage-times are independent and exponentially distributed with parameters $\lambda_1, \ldots, \lambda_{m-1}$, while the bottleneck stage time is exponentially distributed with parameter $\lambda > \max\{\lambda_i\}$. Then we get a statistically significant deviation in behavior only when $\lambda > 1/c =: \tau$ where τ represents the critical threshold and c is the solution of Equation 21. For $\lambda < \tau$ there will be no statistically significant benefit to bottleneck optimization.

7. EXTENSION TO A BROADER CLASS OF DISTRIBUTIONS

So far, our results assume *exponential* stage-time distributions. We now offer several types of evidence that these results hold for a broader class of distributions.

Theoretical considerations. Similar generalizations have been extensively studied in random-matrix theory and are exemplified by the well-known *universality conjecture* [Deift 2007]. This conjecture considers matrix $S_{n,m}$ in Proposition 4.1 and replaces the Gaussian distribution by an arbitrary distribution f_w with the same mean and variance. The claim is that the largest eigenvalue will be described by the same TW_2 distribution, as long as the fourth moment of f_w is bounded. This conjecture is supported by numerical data [Deift 2007], is commonly viewed as a nonlinear law of large numbers for max-eigenvalues, and mirrors what has been recently proven for min-eigenvalues by Tao and Vu [2010]. We state an analogous nonlinear law of large numbers for MEEL.

CONJECTURE 7.1. *Consider two n-stage stochastic streams where stage-time dis-*

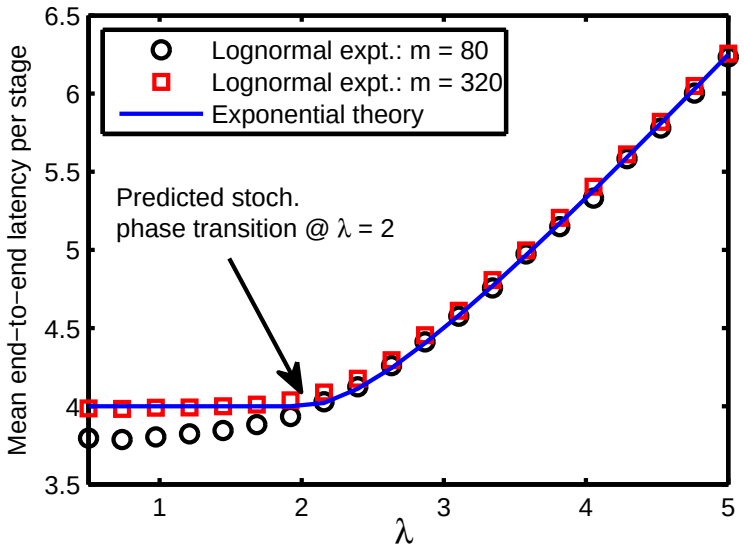

Fig. 8. Theoretical predictions for MEEL with *exponentially* distributed stages and a single bottleneck having a mean stage time of λ compared to simulation results averaged over 1000 independent trials for *log-normally* distributed stages, where the remaining stages have mean stage time of one unit. Equally good fits were produced up to $n = 1000$ (not shown).

tributions are in stochastic order.[7] *The first stream exhibits arbitrary stage distributions with means μ_i, variances σ_i^2 and bounded fourth moments. The second stream exhibits exponential stage-time distributions with parameters $\lambda_i = \sigma_i$ and additional* linear shifts *to adjust their means to match μ_i. Then the two streams exhibit the same cost of stochasticity and the same threshold τ below which improvements to MEEL latency vanish.*[8]

Empirical evidence for normal distributions. Assume $m-1$ stages with mean $\mu = 1$ and variance $i/(n-1)$ at the i-th stage. Let the bottleneck occur at the m-th stage, normally distributed with mean $\mu = 1$ and variance λ^2. The cost of stochasticity can be computed using Formula 28 with $\lambda_i = i/(n-1)$ and predicts experimental results with 5% accuracy. The phase-transition threshold $\tau := 1/c$ where c is predicted by Equation 21 matches empirical results, as seen in Figure 7.
Empirical evidence for log-normal distributions with p.d.f.

$$f(t;\mu,\sigma) = \frac{1}{t\sigma\sqrt{2\pi}} \exp\big(-\frac{(\log t - \mu)^2}{2\sigma^2}\big), \qquad t > 0. \tag{29}$$

We set $\mu = \log(\lambda/\sqrt{2})$, $\sigma = \sqrt{\log 2}$ to match the mean and variance of the exponential distribution with parameter λ. Our earlier predictions are validated in this case by simulation data shown in Figure 8.

[7]For real random variables A and B, $A \leq B$ when $\Pr[A > x] \leq \Pr[B > x]\ \forall x$.
[8]Asymptotic equality neglects distribution-dependent higher-order terms.

8. COMPARING COSTS TO BENEFITS OF STOCHASTICITY

Recall that conclusions can be drawn from Amdahl's law that are relevant to both hardware design and software optimization. In a similar spirit, we now consider *software streams with stage-times* that are randomized even for identical input data. In commercial EDA tool-chains, examples include (*i*) random restarts in leading DPLL-style SAT-solvers, (*ii*) the Fiduccia-Mattheyses heuristic for netlist partitioning used with randomized initial partitions, and (*iii*) the framework of simulated annealing, used in circuit placement and chip floorplanning, where *move selection* during local search is randomized. Numerical EDA algorithms often exhibit very different convergence in different configurations, and trying multiple settings on identical inputs in parallel was shown useful [Dong and Li 2009].

Using additional computational cores can reduce the means of the stochastic stage-times without reworking the algorithms. This is achieved by running multiple independent jobs on identical inputs. Due to stochasticity, some jobs will finish earlier, at which point the other equivalent jobs can be terminated.

LEMMA 8.1. *Let $y_1, \ldots, y_s$ be independent exponentially distributed random variables with mean parameters $\lambda_1, \ldots, \lambda_s$, respectively as in Definition 4.1. Then*

$$\min(y_1, \ldots, y_s),$$

is also exponentially distributed with parameter $1/(1/\lambda_1 + ... + 1/\lambda_s)$.

PROOF. See for example [Ross 2004]. □

COROLLARY 8.1. *Let $y_1, \ldots, y_s$ be s i.i.d. exponentially distributed random variables with mean λ. Then $\min(y_1, \ldots, y_s)$ is exponentially distributed with mean λ/s.*

We can apply this corollary to realize a benefit of stochasticity in the following manner. In the setting of Section 6, consider a single exponentially distributed bottleneck stage with mean λ_m. By the law of vanishing returns, only $\lceil \lambda_m/\tau \rceil < \lceil \lambda_m / \max(\lambda_1, \ldots, \lambda_{m-1}) \rceil$ *identical cores achieve the maximum possible gain*, and no additional independent starts can improve MEEL, despite improving the bottleneck.[9] In Figure 9, this technique is *greedily* applied to two bottlenecks ($\lambda_1 = 15$, $\lambda_2 = 30$). A more effective *balanced allocation* splits s available processors among k bottlenecks as $\Sigma_i^k s_i = s$ so as to minimize $\Sigma_i^k (\lambda_i/s_i)$.

The benefits of stochasticity in software streams can be contrasted with its *costs*. For example, in Figure 5 at $\lambda = 5$ the costs (gaps between solid and dashed lines) are small, but the benefits can produce a net $2\times$ reduction in MEEL.

9. CONCLUSIONS

Our work establishes a far-reaching connection between (*i*) the performance evaluation of stream processing and (*ii*) the *spectral theory of random matrices* [Johansson 2000; Johnstone 2001; Edelman and Rao 2005; Deift 2007; Tao and Vu 2010]. The analytical models we derived for the costs of stochasticity in stream processing are confirmed by numerical simulations with high accuracy and exhibit previously unknown scaling trends, such as a *a law of vanishing returns*. To the best of our

[9]Our analysis neglects higher-order terms. Empirically, a very small improvement may be observed, as in Figures 6 and 8.

knowledge, relevant results from *queuing theory* Glynn and Whitt [1991] only cover the case of balanced streams, and only to the first order. In contrast, our analytical predictions agree with empirical data for both balanced and unbalanced stochastic streams with several types of stage-time distributions, where only the mean and the variance seem to affect key parameters of interest. In the random-matrix setting [Nadakuditi and Silverstein 2010], it has been theoretically established that correlations only affect (negligible) higher-order terms.

We have outlined an optimization approach for allocating parallel cores to speed-up bottlenecks in stochastic software streams. In this context, we illustrate how the benefits of stochastic runtimes may outweigh their adverse impact on end-to-end latency of stream processors. Our analysis is analogous to Amdahl's law in that we quantify the sensitivity of the overall performance to one bottleneck task. Like in Amdahl's law, all tasks are sequentially ordered. The key difference, however, is that our tasks work on streaming data, which allows a greater degree of parallelism despite sequential constraints. As a result, the overall performance can be a lot less sensitive to the greatest bottleneck. The trend itself may have been expected, but we demonstrate that the dependence undergoes a *phase transition*, which can hardly be anticipated by conventional intuition.

Our work focused on analytical estimates and algorithmically-simple techniques for bottleneck detection. This, in particular, limited our applications to the simple *linear* task dependency graph. However, our technique can undoubtedly be extended to a broader range of topologies. When the number of directed path in a given topology is relatively small, our technique can be applied to each path. In cases with significant path reconvergence (a common reason for exponential explosion in path counts), groups of tasks can be merged into larger tasks, reducing path counts. Such extensions are the subject of our ongoing work.

Application domains for our results are diverse and include power-aware pipeline scheduling [Ghasemazar and Pedram 2011], energy-proportional computing [Cameron 2010; Ryckbosch et al. 2011], stochastic analysis and optimization of NoCs [Bogdan and Marculescu 2009; 2011b; 2010], simulation and optimization of many-core CPUs [Burke et al. 2008], scheduling scientific workflows on supercomputers [Thibodeau 2009; Tan 2010], and various applications of stochastic task graphs outlined in the book [Manolache et al. 2007b] to list a few [Kerbyson et al. 2011].

Acknowledgements

R.R.N thanks Jinho Baik for many insights and stimulating discussions. This work was partially supported by NSF CCF-1116115.

REFERENCES

Agrawal, K., He, Y., and Leiserson, C. 2006. An empirical evaluation of work stealing with parallelism feedback. In *ICDCS*. Citeseer.

Amdahl, G. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. April 18-20, 1967, Spring Joint Computer Conf.* ACM, 483–485.

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., et al. 2009. A view of the parallel computing landscape. *Communications of the ACM 52,* 10, 56–67.

Baik, J., Ben Arous, G., and Péché, S. 2005. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *Annals of Probability*, 1643–1697.

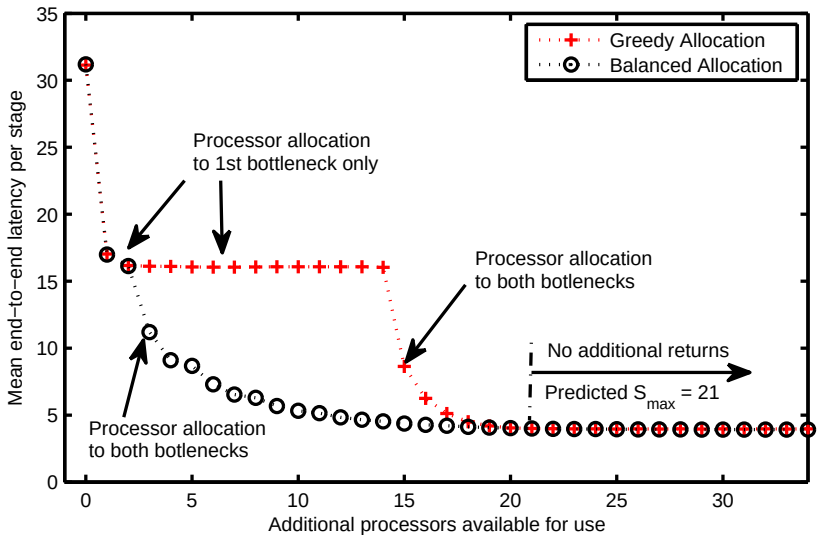

Fig. 9. Two strategies for processor allocation in a two-stage stochastic stream with $\lambda_1 = 15$ and $\lambda_2 = 30$.

Baik, J., Deift, P., and Johansson, K. 1999. On the distribution of the length of the longest increasing subsequence of random permutations. *Journal of the American Mathematical Society 12,* 4, 1119–1178.

Bogdan, P. and Marculescu, R. 2009. Statistical physics approaches for network-on-chip traffic characterization. In *Proc. IEEE/ACM Int'l Conf. Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. ACM, 461–470.

Bogdan, P. and Marculescu, R. 2010. Workload characterization and its impact on multicore platform design. In *Proc. IEEE/ACM/IFIP Int'l Conf. Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. ACM, 231–240.

Bogdan, P. and Marculescu, R. 2011a. Cyberphysical systems: workload modeling and design optimization. *Design & Test of Computers, IEEE 28,* 4, 78–87.

Bogdan, P. and Marculescu, R. 2011b. Non-stationary traffic analysis and its implications on multicore platform design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 30,* 4, 508–519.

Bornemann, F. 2010. On the numerical evaluation of distributions in random matrix theory: A review. *Markov Processes Relat. Fields 16*, 803–866.

Borodin, A. and Péché, S. 2008. Airy kernel with two sets of parameters in directed percolation and random matrix theory. *Journal of Statistical Physics 132,* 2, 275–290.

Burke, D., Wawrzynek, J., Asanovic, K., Krasnov, A., Schultz, A., Gibeling, G., and Droz, P. 2008. Ramp blue: Implementation of a manycore 1008 processor system. *Reconfig. Sys. Summer Inst.(RSSI)*.

Cameron, K. 2010. The Challenges of Energy-Proportional Computing. *IEEE Computer 43,* 5, 82–83.

Chrysos, G., Dean, J., Hicks, J., Waldspurger, C., and Weihl, W. 1998. Method for estimating statistics of properties of instructions processed by a processor pipeline. US Patent 5,809,450.

Cover, T. and Thomas, J. 2006. *Elements of information theory.* Wiley.

Davare, A., Zhu, Q., Di Natale, M., Pinello, C., Kanajan, S., and Sangiovanni-Vincentelli,

A. Period optimization for hard real-time distributed automotive systems. In *Proc. ACM/IEEE Design Automation Conf. (DAC)*.

Deift, P. 2007. Universality for mathematical and physical systems. In *Int'l Congress of Mathematicians. Vol. I*. Eur. Math. Soc., Zürich, 125–152.

Dong, W. and Li, P. 2009. Parallelizable stable explicit numerical integration for efficient circuit simulation. In *Proc. ACM/IEEE Design Automation Conf. (DAC)*. IEEE, 382–385.

Edelman, A. and Rao, N. 2005. Random matrix theory. *Acta Numerica 14,* 233-297, 139.

El Karoui, N. 2007. Tracy-Widom limit for the largest eigenvalue of a large class of complex sample covariance matrices. *The Annals of Probability 35,* 2, 663–714.

Ghasemazar, M. and Pedram, M. 2011. Optimizing the Power-Delay Product of a Linear Pipeine by Opportunistic Time Borrowing. *IEEE Trans. on CAD 30,* 10, 1493–1506.

Glynn, P. and Whitt, W. 1991. Departures from many queues in series. *The Annals of Applied Probability 1,* 4, 546–572.

Hill, M. and Marty, M. 2008. Amdahl's law in the multicore era. *Computer 41,* 7, 33–38.

Johansson, K. 2000. Shape fluctuations and random matrices. *Communications in Mathematical Physics 209,* 2, 437–476.

Johnstone, I. 2001. On the distribution of the largest eigenvalue in principal components analysis. *Annals of Statistics*, 295–327.

Kerbyson, D. J., Vishnu, A., Barker, K. J., and Hoisie, A. 2011. Codesign challenges for exascale systems: Performance, power, and reliability. *IEEE Computer 44,* 11, 37–43.

Lipman, J. and Stout, Q. 2006. A performance analysis of local synchronization. In *Proc. ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 260.

Manolache, S., Eles, P., and Peng, Z. 2007a. Fault and energy-aware communication mapping with guaranteed latency. *Int'l fJournal of Parallel Programming 35,* 2, 125–156.

Manolache, S., Eles, P., and Peng, Z. 2007b. Real-time applications with stochastic task execution times - analysis and optimisation. Springer, 1–152.

Nadakuditi, R. and Benaych-Georges, F. 2010. The breakdown point of signal subspace estimation. In *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 177–180.

Nadakuditi, R. and Silverstein, J. 2010. Fundamental limit of sample generalized eigenvalue based detection of signals in noise using relatively few signal-bearing and noise-only samples. *IEEE Journal of Selected Topics in Signal Processing 4,* 3, 468–480.

Ogras, U. and Marculescu, R. 2005. Application-specific network-on-chip architecture customization via long-range link insertion. In *IEEE/ACM Int'l Conf. Computer-Aided Design (ICCAD)*. IEEE, 246–253.

Rajsbaum, S. and Sidi, M. 1994. On the performance of synchronized programs in distributed networks with random processing times and transmission delays. *IEEE Trans. on Parallel and Distributed Systems 5,* 9, 939–950.

Ross, S. 2004. *Introduction to probability and statistics for engineers and scientists*. Acad. Press.

Ryckbosch, F., Polfliet, S., and Eeckhout, L. 2011. Trends in Server Energy Proportionality. *IEEE Computer 44,* 9, 69–72.

Srinivasan, R. 1993. Queues in series via interacting particle systems. *Mathematics of Operations Research*, 39–50.

Tan, W. 2010. Network Analysis of Scientific Workflows: A Gateway to Reuse. *IEEE Computer 43,* 10, 54–61.

Tao, T. and Vu, V. 2010. Random matrices: Universality of local eigenvalue statistics up to the edge. *Communications in Mathematical Physics 298,* 2, 549–572.

Tembe, S. and Wolff, R. 1974. The optimal order of service in tandem queues. *Operations Research*, 824–832.

Thibodeau, P. 2009. Supercomputers with 100 million cores coming by 2018. *Computerworld 11,* 16, 09.

Received December 2011; revised May 2012; accepted ?? ????