CCEval: Accurately and Confidently Evaluating Performance Metrics of Congestion Control Algorithms for Datacenter Networks

Tianfeng Liu[1], Kaihui Gao[1], Li Chen[1], Dan Li[2], Jin Guang[3], Xinyun Chen[3],
Vincent Liu[4], Zhiyong Chen[2], Yiwei Zhang[2], Ni Jin[1,5], Ran Zhang[1]
[1]*Zhongguancun Laboratory,* [2]*Tsinghua University,* [3]*The Chinese University of Hong Kong, Shenzhen,*
[4]*University of Pennsylvania,* [5]*Beijing University of Posts and Telecommunications*

Abstract

Congestion control in datacenter networks (DCNs) is a highly active research area. Typical CCA evaluation workflows contain three steps: generate experimental configurations, execute the experiments, and estimate performance metrics using results from multiple trials. However, due to variability brought by random traffic workloads and single-digit trial counts, common experimental methodologies fail to provide enough confidence to properly evaluate CCA performance.

We propose CCEval, an evaluation framework for accurately and confidently estimating performance metrics of CCAs in DCNs. The key idea is using *confidence intervals* and more trials to quantify and improve the accuracy and confidence of performance metrics. To this end, we propose a model-free estimation algorithm to calculate the confidence intervals and forecast the required trial count for a given accuracy, confidence level, metric, and CCA. We further design a model-based tail quantile estimation algorithm to reduce the needed trial counts significantly without losing accuracy and confidence. Extensive experiments on simulators and real-world testbeds with four CCAs on typical topologies and flow distributions show that CCEval can produce estimations of performance metrics accurately and confidently, with 1% relative margin of error and 95% confidence level, and can reduce trial counts by 75%∼80% for tail quantile estimation.

1 Introduction

Congestion control has been an active area of research since the first indications of congestive collapse in the 1980s [26]. With the rise of data-intensive applications, including, most recently, LLMs and distributed training [1, 29, 32, 36], numerous congestion control algorithms (CCAs) focused on datacenter networks (DCNs) have been proposed, including DCTCP [3], DCQCN [54], TIMELY [35], HPCC [30], and others. For newly proposed CCAs, researchers should ensure that algorithms are evaluated fairly against existing CCAs across a wide range of network scenarios [49]. They should be confident that these performance results are representative of different deployment environments and robust to workload randomness.

The typical DCN CCA evaluation workflow comprises three steps: *(1) Generate the set of experimental configurations* that will be used to evaluate the target CCA, including the network configuration, CCA parameters, and workload characteristics (*e.g.*, flow size distribution and flow arrival distribution) [28, 53]. *(2) Execute the CCA experiments* by repeatedly sampling from the traffic distributions and obtaining many different traffic workloads. Following typical captured and synthetic trace formats [3, 30, 40, 54], workloads are defined by a sequence of flow demands, each with a flow size and start time. For each workload, we execute one *trial* of the CCA experiment on a physical testbed [11, 30, 54], emulator [31], or simulator [15, 20, 38, 48, 52] and collect the experimental results. *(3) Estimate the performance metrics* by calculating statistics based on the results of the trials.

In an ideal world, the CCA would be evaluated over *all* possible workloads [24]. Of course, practical constraints make this impossible—instead, researchers must settle for empirical results, which means there is inevitable error between the empirical and 'true' observations. The gap can be quantified with the help of metrics like standard deviation, standard error, or confidence intervals. Particularly for standard deviation, it is also important to consider the trial count when evaluating the reliability of the results.

Unfortunately, we find that most recent networking papers (the authors' included) take a relatively unprincipled approach to evaluating statistical significance, which can lead to misleading conclusions. In fact, our survey (see §2.2.1) of 20 recent, highly cited CCA papers from top conferences shows that 85% fail to report error bars and/or trial count entirely; the remainder only report standard deviation using very limited trials ($\leq$10). In §2.2.2, we conduct our own experiments on these protocols and find large variances in performance for different random traffic seeds (*e.g.*, standard deviation is up to 50% of the mean). As a concrete example of the impact of this variability, we compare median FCTs of large flows using

DCQCN and one of its follow-ups, HPCC. With only 10 trials, we found that HPCC underperformed DCQCN; however, after 500 trials, we arrived at the opposite conclusion. We emphasize that our results do not invalidate the contributions or findings of the papers we survey. Indeed, the intellectual and commercial impact of these protocols is better proof than any graph. Rather, we argue that methodological improvements can improve our ability to evaluate CCAs that the community and production traffic have not yet vetted.

To that end, we present an evaluation framework capable of accurately and confidently estimating CCA performance and contend that such a framework should be able to:

- **R1 Help users quantify the accuracy and confidence of their experimental results.** The framework should be able to quantify the accuracy (*i.e.*, the margin of error or confidence interval) and confidence (probability of containing the true statistics, *i.e.*, confidence level) of experimental results despite the high variability of many modern CCAs.
- **R2 Forecast the required trial count for a given confidence interval and level.** As constraints on resources and time may constrain the feasible number of trials, the system should be able to forecast the required trial count for a given confidence interval and level. Researchers can use this to better allocate resources and set expectations.
- **R3 Generalize to any CCAs.** The framework should be general to any CCA without requiring knowledge of the CCA's details or manual modeling efforts for new CCAs.
- **R4 Reduce the necessary trial count without sacrificing accuracy or confidence.** Due to the need for many trials, CCA experiments have long runtimes, which may not be affordable to researchers. The framework should reduce runtime by significantly reducing the required trials without losing accuracy and confidence.

In this paper, we propose CCEval, a framework for accurately and confidently estimating performance metrics of congestion control algorithms in datacenter networks.

To meet R1, we propose to explicitly compute the *confidence interval* [39] of the estimated performance metrics. The confidence interval is an *interval estimation* [13] with an interval width and confidence level. The interval width is the accuracy (or error) between the true (but unknown) value and the estimated value. The confidence level is the probability that the true value is contained in the interval. By conducting more trials, researchers can reduce the confidence interval and increase the confidence level of their performance results.

To meet R2 and R3, we propose a novel model-free estimation algorithm. This algorithm leverages the *independent and identically distributed (IID)* property of running multiple CCA trials, and uses the Monte Carlo method [42] to construct the confidence interval of the estimated metrics. "Model-free" means that this algorithm only relies on the independent and identically distributed assumption and can apply to any CCAs and any performance metrics. Another advantage of this algorithm is that it can accurately forecast the required trial counts for a given accuracy and confidence level based on comparatively few initial trials.

To improve on the above and meet R4, we propose a model-based *tail quantile* estimation algorithm to reduce the trial count without losing accuracy and confidence for tail quantile estimation. We leverage the *importance sampling (IS)* [46] to increase the probability of rare CC events (such as events that cause long-tail performance) by using an IS distribution. As a result, this algorithm can use fewer trials to achieve the same accuracy and confidence as the model-free algorithm. We further propose an adaptive cross-entropy algorithm to search for near-optimal IS distributions efficiently.

We implement and evaluate CCEval on top of NS-3 [38], M3 [28], Parsimon [53], and a real-world testbed. We perform extensive experiments in estimating four different quantiles of FCT using four widely used CCAs (DCTCP [3], DCQCN [54], TIMELY [35], HPCC [30]) under four publicly available flow distributions and two different DCN topologies (DumbBell and FatTree). The results show that the model-free estimation algorithm can estimate performance metrics with 1% relative margin of error and 95% confidence level for all experimental settings. Further, this algorithm can forecast the number of trials for a given accuracy and confidence level a priori, with errors as low as 1%. Our model-based estimation tail quantile algorithm can reduce trial count by 75%∼80% while converging to the same accuracy and confidence level as the model-free algorithm. We will open-source CCEval in https://github.com/NASP-THU/CCEval.

2 Background and Motivation

2.1 Typical CCA Evaluation Workflow

Congestion control in DCNs is an active research area. Many CCAs have been proposed recently, as modern distributed applications require a high-performance transport layer to accelerate data transmission. In both academia and industry (*e.g.*, Google Cloud [11], Microsoft Azure [54], Alibaba Cloud [30]), the typical evaluation workflow for CCAs contains three steps, as shown in Figure 1.

(1) Generate the set of experimental configurations. The first step is to enumerate the range of experimental configurations that will be used to evaluate the target CCA. The configuration encompasses three main components. The first is the workload configuration, which often specifies the flow size distribution (*e.g.*, from some empirical distribution [30, 40, 54]), flow arrival distribution (*e.g.*, an exponential distribution), communication pattern (*e.g.*, all-to-all traffic [30] or incast [16, 54]), traffic duration, and traffic load percentage. The second is the network configuration, including but not limited to the network topology, number of hosts, buffer size, PFC settings, and link bandwidth. The third is the

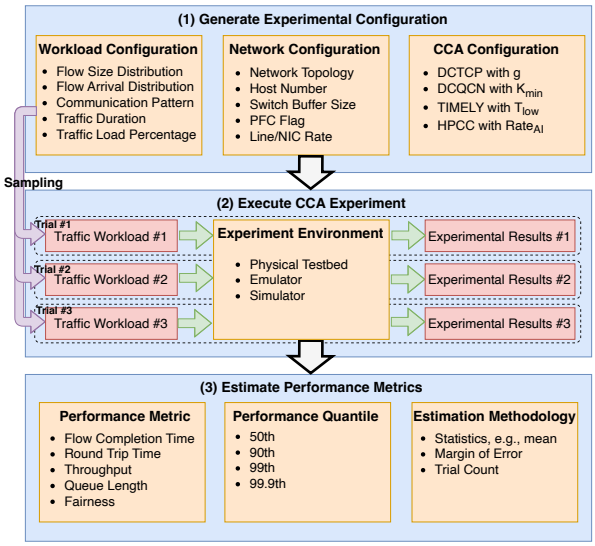


Figure 1: Typical evaluation workflow for CCAs in DCNs.

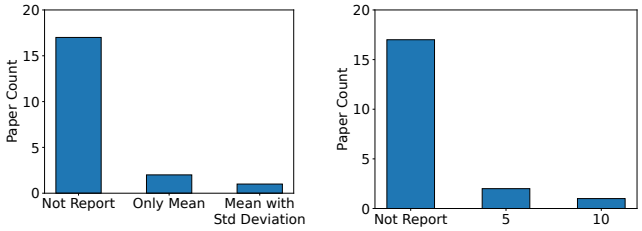
CCA configuration options, *e.g.*, K_{min} in DCQCN.

(2) Execute the CCA experiments. Using the above configurations, we repeatedly sample from the traffic distributions to obtain many different workload traces. The advantage of using traffic distributions rather than a particular trace is precisely this ability to evaluate the CCA over an entire class of traffic patterns rather than a single run. For CCA evaluations, the traces are typically of flow-level demand and take the form of a sequence of flows, each with a flow size and arrival time. For each trace, researchers conduct experiments on one of three different types of environments: physical testbeds [8, 11, 30, 54], emulators [21, 22, 31] or simulators [15, 20, 38, 48, 50, 52]. We call a single run with a single generated workload trace as one *trial* and the collection of multiple trials an *evaluation*.

(3) Estimate the performance metrics. After conducting experiments, we can estimate performance metrics of the CCA under the tested configurations. To do so, we select a target performance metric, *e.g.*, flow completion time (FCT), round-trip time (RTT), throughput, queue length, etc. As every trial involves many measurements (*e.g.*, multiple flows or multiple samples of the queue length for different devices over time) we are generally interested in the distributional qualities of the measured results, *e.g.*, the 50th, 99th, 99.9th, etc. quantiles. As different trials can produce different results due to the variability of the workloads and CCA, we must also aggregate the results of different trials. For this, the most common method is to report the mean over the trials (*e.g.*, the mean of the 99th quantile over all trials) as it is a measure of the central tendency of the data, giving a typical or average outcome across all the trials.

2.2 Confidence in Recent CCA Evaluations

The true mean of the trial statistics is, of course, impossible to measure directly as doing so would require us to run trials over all possible workloads. Instead, the experimentalists leverage confidence intervals to quantify uncertainty in the relationship between the observed mean and the true mean [24]. Unfortunately, in our systematic literature survey, we find that most recent CCA work (even work that has been highly influential) fails to report these confidence intervals and in many cases even fails to specify the complete experimental methodology. Our simulation experiments further demonstrate the potentially misleading nature of naive methodologies.



(a) Mean with margin of error (b) Trial count

Figure 2: Statistical results of estimation methodology used in recent research papers on CCAs in DCNs.

2.2.1 Literature Survey and Findings

Our literature survey covers 20 highly-cited DCN CCA papers (7,606 citations in total) selected from top conferences (*e.g.*, SIGCOMM, NSDI, ATC, and EuroSys) in recent years. For each paper, we try our best to analyze two types of criteria: (i) reporting the mean with margin of error, (ii) reporting the trial count used in each experiment. Figure 2 summarizes the results of our survey, which we distill into the following:

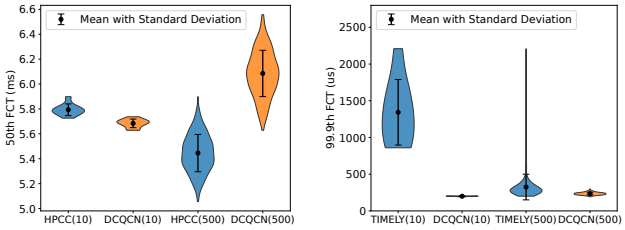
(1) Most papers fail to report margin of error and trial count. We find 17 papers (85%) do not report error or trial counts at all. Only one reports the standard deviation among trials.

(2) Most papers use a very small number of trials. Only three papers report their trial counts, but all use a very limited number of trials. More specifically, two papers conducted 5 trials per datapoint, and only Pantheon [49] performed 10 trials.

2.2.2 Experimental Result and Observations

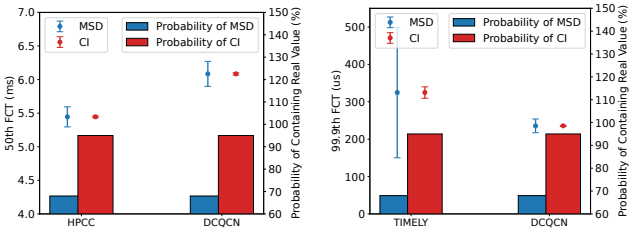
We evaluate the impact of variability using four CCAs (HPCC, DCQCN, TIMELY, and DCTCP), and execute them on NS-3 [38] using a FatTree topology ($k = 8$) with 30% average network load and WebSearch flow distribution. In each trial, we set the experiment duration to 0.1 seconds (durations are discussed in §6.4), corresponding to ∼30,000 flows per trial. For each trial, we sort all flows by FCT and report the 50th, 99th, and 99.9th quantiles for small and large flows separately.

In Figure 3, we use the violin plot [23] to show the distribution of different quantiles. In the violin plot, we also report the mean value with standard deviation (black error bar), which is used in current evaluation workflows. We also tried other performance metrics, such as FCT slowdown [30], which have similar trends. Based on these results, we make



(a) 50th FCT of large flows. (b) 99.9th FCT of small flows.

Figure 3: FCT quantile distributions of large flows (flow size > 10MB) and small flows (< 10KB), running on a FatTree with WebSearch. The number in (), *e.g.*, 10, 500, is the trial count.



(a) 50th FCT of large flows. (b) 99.9th FCT of small flows.

Figure 4: The margin of error and probability of containing true mean value, using standard deviation (MSD) versus confidence intervals (CIs), running 500 trials on FatTree with WebSearch. CI has smaller error and a higher probability of containing the true mean.

the following observations, coming from system variance and statistical uncertainty respectively:

(1) Due to the system variance, variability of FCT quantiles is inevitable and is largely independent of the number of trials. Due to the system variance (*e.g.*, the randomness of different traffic workloads and the complexity of flow interactions), there is inevitable variability of FCT quantiles, and the observed variability is independent of the number of trials (sometimes going up as we observe more trials), sometimes reaching up to 50% of the mean value. As these examples and the ones in Figure 4 illustrate, *standard deviation*, while useful for some purposes, does not describe our confidence in whether we have found the true mean—only that the range has ∼68% probability of containing the true mean in the range of one standard deviation.

(2) Limited trial counts lead to low confidence and misleading conclusions for both large and small flows. Due to limited trial counts, the variability of the performance metric may be *overestimated* or *underestimated*, and there is no way to predict which occurred without running more trials. As shown in Figure 3a, with only 10 trials, we might conclude that the 50th FCT of HPCC is larger than that of DCQCN. However, after 500 trials, the conclusion flips. A similar phenomenon also occurs for small flows in Figure 3b. If using only 10 trials, the 99.9th FCT of TIMELY is 600% larger than that of DCQCN. After using 500 trials, the 99.9th FCT of TIMELY is only 50% larger than that of DCQCN.

2.3 Requirements for Confident Estimation

We argue that an evaluation framework capable of confidently estimating the performance metrics of CCAs must meet the following four requirements.

R1: Help users quantify the accuracy and confidence of their experimental results. The framework should help measure and quantify the true population mean of the target performance metrics, and be able to quantify the accuracy (confidence interval) and certainty (confidence level) of the results. Unlike prior CCA evaluation frameworks [49], we focus explicitly on confidence intervals/levels rather than only the Mean and Standard Deviation (MSD).

R2: Forecast the required trial count for a given accuracy and confidence level. As constraints on resources and time may constrain the feasible number of trials, the system should be able to forecast the required trial count for a given accuracy and confidence level. These forecasts can be used to better allocate resources (*e.g.*, planning the number of experiments, configurations, and datapoints) and set expectations about achievable target confidence levels. We note that some works [34, 47] propose calculating confidence intervals using nonparametric analysis [10]. However, this method does not assume anything about the underlying probability distribution, hence, they struggle to forecast the required trial count.

R3: Generalize to any CCAs. This framework should be general to any CCA without constraining the format or features of the CCA and without detailed knowledge of the CCAs inner workings. This improves robustness and removes the burden of manually modeling new CCAs.

R4: Reduce the necessary trial count without sacrificing accuracy or confidence. As we move deeper into analysis, we find that accurate evaluations with high confidence require long runtimes due to very large trial counts. Hence, we need an algorithm that can reduce trial counts and experiment runtimes without losing result accuracy and confidence.

3 System Overview

Figure 5 shows the architecture and workflow of CCEval.

Input & output. CCEval takes two configurations as input. The first is an experimental configuration identical to current evaluation workflows as described in §2.1. The second is the estimation configuration introduced by CCEval, including the performance metric with the corresponding quantile, the required accuracy, and the target confidence level. Based on these inputs, the output is the estimated mean and confidence interval for the performance metrics.

Main components and workflow. As shown in Figure 5, CCEval contains three main components:

- A traffic distribution sampler.
- An experiment executor.
- A performance metric estimator.

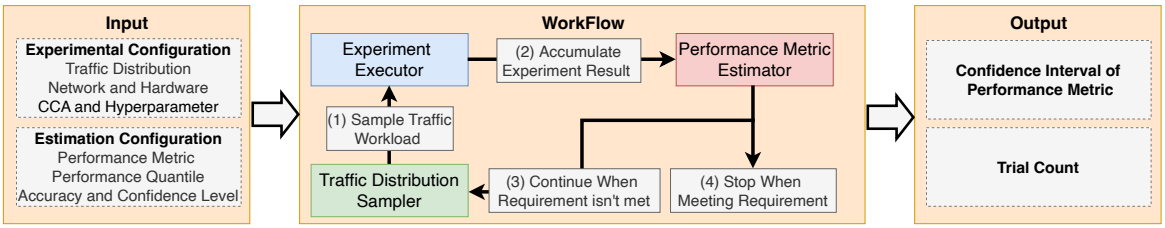


Figure 5: Architecture and workflow of CCEval.

The traffic distribution sampler samples new traffic workloads from the traffic distribution. The experiment executor executes trials based on the experiment configuration and generates experimental results. The experiment executor can be executed on physical testbeds, simulators [15, 25, 38], or emulators [31]. Based on the results of the cumulative trials up to that point, the performance metric estimator executes the estimation algorithm and calculates the confidence interval for all of the target performance metrics.

If, after the current iteration, the required accuracy or confidence level is not met, CCEval repeats the process and goes back to step (1). After only a few such iterations (default of 20), the system is able to output an estimate of the remaining number of trials until convergence with very high accuracy, following the technique described in §4.2. When the target confidence level is met, the system terminates and outputs the desired metrics.

4 Accurate and Confident Estimation

In this section, we propose to obtain an accurate and confident estimation of performance metrics by calculating the confidence interval using parametric analysis.

4.1 Estimation using Confidence Interval

4.1.1 Confidence Interval (CI)

We propose to use the confidence interval to quantify the accuracy and confidence of estimated performance metrics. The definition of a confidence interval [39] is as follows.

Definition. *Suppose that there is a random variable Z with an unknown parameter θ. A random interval $I_\rho = (\theta_1, \theta_2)$ is a confidence interval (CI) with confidence level $1-\rho$ if and only if $P\{\theta_1 \le \theta \le \theta_2\} = 1-\rho$. In other words, the probability of the true value of θ in the random interval (θ_1, θ_2) is $1-\rho$. θ_1 is the lower bound of the CI, θ_2 is the upper bound of CI.*

The half-width $(\theta_2 - \theta_1)/2$ of CI is defined as the *accuracy* of CI, which measures the error between the real value θ and the empirical value $\hat{\theta}$. If the confidence interval has high accuracy and high confidence level, we can say that we get an *accurate and confident* estimation of θ.

4.1.2 Confidence Interval of Performance Metrics

We propose using the *Monte Carlo method* [42] to construct the confidence interval of performance metrics.

After conducting multiple trials of CCA experiments, the performance metrics thar researcher interested in are a type of random variable Z and we want to compute the mean value $z = \mathbb{E}Z$. Obviously, there is no analytical solution of z and it can be simulated using the Monte Carlo method. By conducting N trials, we have N samples of Z: $Z_1, \ldots, Z_N$, and all samples are *independent and identically distributed (IID)*. Hence, z can be estimated by the empirical mean $\hat{z} = \frac{Z_1 + \cdots + Z_N}{N}$. Assuming $\sigma^2 = \mathbb{V}arZ < \infty$, based on the central limit theorem (CLT) [4], the distribution of $\hat{z}$ can be approximated by $\hat{z} \to z + \frac{\sigma V}{\sqrt{N}}$, as $N \to \infty$, where $V \sim \mathcal{N}(0,1)$ is the standard normal distribution. Suppose v_ρ is the ρ-quantile of the standard normal distribution, i.e., $\Phi(v_\rho) = \rho$, the random interval I_ρ $(\hat{z} - \frac{v_{1-\rho/2}\cdot\sigma}{\sqrt{N}}, \hat{z} - \frac{v_{\rho/2}\cdot\sigma}{\sqrt{N}})$ is the confidence interval of z with confidence level $1-\rho$. Since the real σ^2 is unknown as well, it can be estimated using the empirical variance $\hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Z_i - \hat{z})^2$. Using the empirical variance, the confidence interval of z with $1-\rho$ confidence level is estimated by

$$(\hat{z} - \frac{v_{1-\rho/2}\cdot\hat{\sigma}_N}{\sqrt{N}}, \hat{z} - \frac{v_{\rho/2}\cdot\hat{\sigma}_N}{\sqrt{N}}) = \hat{z} \pm \frac{v_{1-\rho/2}\cdot\hat{\sigma}_N}{\sqrt{N}} \quad (1)$$

4.2 Model-free Performance Estimation

We propose a model-free estimation algorithm to generate the confidence interval of performance metrics with a given accuracy and confidence level.

Based on Equation 1, for given relative accuracy $\varepsilon|\hat{z}|$ and the confidence level $1-\rho$, we need to choose the trial count N, such that the convergence condition

$$v_{1-\rho/2}\cdot\hat{\sigma}_N \le \varepsilon|\hat{z}|\sqrt{N} \quad (2)$$

is met, where $\hat{\sigma}_N$ is the empirical standard deviation of N trials.

Based on the above idea, the model-free algorithm contains two stages: an exploratory stage and a production stage.

(1) Exploratory stage: In this stage, we first run N trials of the CC experiment and calculate the empirical variance. Since the empirical variance is highly unreliable with a small number

of trials, we should use some moderately large number (such as $N = 20$ [4]) to estimate the empirical variance.

(2) Production stage: In this stage, we run more trials. After getting the results from both the exploratory and production stages, we recalculate the empirical variance and compare the convergence condition in Equation 2. If the convergence condition is met, we stop running any more trials; otherwise, we continue until convergence.

Convergence rate. Based on the convergence condition, the convergence rate of this algorithm is $\sqrt{N}$. If we want to increase the accuracy of the confidence interval by one order of magnitude, we need to increase the trial count, N, by a factor of 100. Hence, if we want to get an accurate estimation of performance metrics (such as 0.01%), we need upwards of millions or billions of trials!

Estimating the number of trials. Equation 1 also suggests that, given the relative error ε and confidence level $1-\rho$, we can roughly estimate the trial count N_e we should conduct:

$$N_e \approx \frac{z^2_{1-\rho/2}\hat{\sigma}^2_N}{\varepsilon^2 \hat{z}^2} \tag{3}$$

Hence, researchers can easily know how many trials they need by only conducting a few exploratory runs.

4.3 Model-based Tail Quantile Estimation

Clearly, the model-free estimation algorithm still needs a large trial count to achieve accurate estimation with high confidence. We propose a novel model-related tail quantile algorithm to significantly reduce the number of trials while maintaining the same accuracy and confidence of *tail quantile* performance.

4.3.1 Algorithm Overview

The model-based estimation algorithm contains five logical parts, as shown in Figure 6.

Firstly, in §4.3.2, we model the input distribution (inter-arrival time and service time) and the output distribution (response time of each flow) considering the unique property of flow distribution and impacts of CCA on the network.

Secondly, in §4.3.3, since every trial is independent of each another, multiple trials can be treated as a *regenerative process* [43]. We propose to use the *renewal reward theorem* [12] to estimate the tail probability and tail quantile.

Thirdly, in §4.3.4, we find that due to the rarity of the events that contribute to the tail quantile, accurate and confident estimation of the tail quantile needs a large trial count. We propose using *importance sampling* [46], which uses a carefully selected probability distribution (a.k.a. *IS distribution*), instead of the original distribution, to make the event of tail quantile occur more frequently. As a result, the estimation algorithm can achieve accurate and confident estimation with fewer trials.

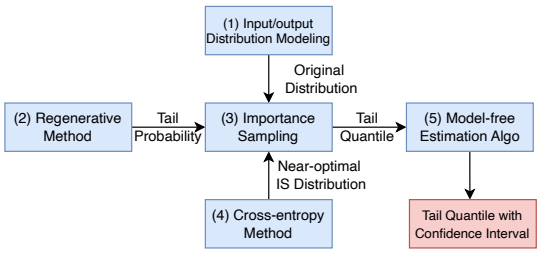


Figure 6: Logical flow of the model-based estimation algorithm.

Fourthly, in §4.3.5, we propose a novel adaptive cross-entropy algorithm to search for the near-optimal IS distribution, which can further reduce the variance brought about by importance sampling and reduce the required trial count.

Finally, we apply the model-free estimation algorithm (§4.2) to generate a confidence interval for the tail quantile of the target performance metrics.

4.3.2 Input and Output Distribution Modeling

Similar to queuing theory [44], after fixing the topology and using the CC algorithm, the entire network is driven by two types of random variables for each flow: *inter-arrival time A* and *service time S*. Hence, we need to model the probability distributions of inter-arrival time f_A and service time f_S.

The inter-arrival time A is the time between the start time of consecutive flows. To model the probability distribution of inter-arrival time, we use the *exponential distribution* with the rate parameter λ [5, 6, 30]. The probability density function of inter-arrival time is: $f_A(x;\lambda) = \lambda e^{-\lambda x}$.

The service time S is the transmission time of each flow. If we fix the network bandwidth (which is generally true in CC experiments), the service time is determined by the size of each flow. We find *the flow size distribution from publicly available sources is both long-tailed and heavy-tailed* [7]. Hence, to model the probability of service time, we use the *logarithmic normal* distribution with the mean parameter μ and the standard variance parameter σ. The probability density function of the service time of each flow is: $f_S(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}x\sigma} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$.

Finally, we want to model the output distribution of the *response time R* (a.k.a. the flow completion time) of all flows in the CC experiment. We choose to model the response time because *this is a general metric that we can estimate the performance of general CCAs without knowledge of algorithm details.* The reason comes from the three parts of response time: $R = T_{rtt} + T_{trans} + T_{queue}$. When using a particular network topology, round trip time T_{rtt} is fixed. When given a flow size, the transmission time of each flow T_{trans} is also fixed. Hence, CCAs only influence the queuing time T_{queue}.

4.3.3 Tail Quantile Estimation

Because the quantile function $Q(p)$ is the inverse function of the cumulative distribution function (CDF), we firstly estimate the tail probability $P\{R > \gamma\}$ for a given response time γ and then calculate the quantile $Q(p)$ based on the tail probability.

Firstly, we give the definition of *regenerative process* [43]:

Definition. *A stochastic process $\{X(t)\}_{t\geq 0}$ is a regenerative process if and only if there exists a sequence of random times $0 = \tau_0 < \tau_1 < \tau_2 < \ldots$, called regeneration times, such that for each $n \geq 0$, the post-regeneration process $\{X(\tau_n + t)\}_{t\geq 0}$ is independent of $\{X(t)\}_{0\leq t<\tau_n}$ and has the same distribution as $\{X(t)\}_{t\geq 0}$. In other words, the stochastic process $\{X(t)\}_{t\geq 0}$ restarts probabilistically at each regeneration time τ_n. The process during the interval $[\tau_n, \tau_{n+1})$, denoted as $\{X(t)\}_{\tau_n\leq t<\tau_{n+1}}$, is called a regenerative cycle.*

We find that *multiple CC trials can be treated as a regenerative process.* The reason is that although the network is busy and generally does not go back to an idle state during any individual trial, multiple trials are *independent* and can be treated as a stochastic process. The process restarts at some random time τ, which is the duration of each trial defined in the experimental methodology.

Based on the assumption of a regenerative process, we use the renewal reward theorem [12] to estimate the tail probability of given response time γ:

$$P(R > \gamma) = \frac{\mathbb{E}\left[\sum_{n=1}^{\alpha} \mathbb{1}\{R_n > \gamma\}\right]}{\mathbb{E}[\alpha]}, \tag{4}$$

where $\mathbb{E}(\cdot)$ is the expectation of a random variable, and for $n = 1, 2, \ldots, \alpha$, R_n represents the response time of n-th flow in this cycle, α is the number of finished flows in this cycle, $\mathbb{1}\{R_n > \gamma\}$ is the indicator function defined as:

$$\mathbb{1}\{R_n > \gamma\} = \begin{cases} 1, & R_n > \gamma \\ 0, & R_n \leq \gamma \end{cases} \tag{5}$$

Based on Equation 4, if we generate m cycles using m trials, the empirical tail probability $\hat{P}(R > \gamma)$ can be estimated by

$$\hat{P}(R > \gamma) = \frac{\sum_{i=1}^{m}\sum_{n=1}^{\alpha_i} \mathbb{1}\{R_{i,n} > \gamma\}}{\sum_{i=1}^{m} \alpha_i}. \tag{6}$$

The empirical tail quantile $\hat{Q}(p)$ can be estimated in the following way. Suppose we generate m cycles with total $\beta = \sum_{i=1}^{m} \alpha_i$ flows and each flow has the corresponding response time R_n. After sorting all R_n in ascending order, we have the ordered samples $\{R_1, \ldots, R_{i'}, \ldots, R_\beta\}$. Then

$$\hat{Q}(p) = \inf\{\gamma : \hat{P}(R > \gamma) < 1 - p\} = R_{n'}, \tag{7}$$

where

$$n' = \min\left\{ n : \sum_{i'=n}^{\beta} 1 \leq (1-p)\sum_{i=1}^{m} \alpha_i \right\}. \tag{8}$$

After estimating the tail quantile, we can use the model-free algorithm to calculate its confidence interval.

4.3.4 Improving Tail Quantile Estimation with IS

However, when γ is very large, the tail event $\{R_n > \gamma\}$ is a rare event; as a result, $\sum_{i=1}^{m}\sum_{n=1}^{\alpha_i} \mathbb{1}\{R_{i,n} > \gamma\}$ in Equation 6 may have large variance and need more trials to estimate it.

In order to reduce the variance, we introduce the *importance sampling (IS)* method, which uses other probability distributions (also called the *IS distributions* $\tilde{f}_A, \tilde{f}_S$) such that the event $\{R_n > \gamma\}$ happens more often than in the original distributions (f_A, f_S). Hence, we could estimate the tail quantile with fewer trials.

After using the IS, the numerator of Equation 4 changes to:

$$\mathbb{E}\left[\sum_{n=1}^{\alpha} \mathbb{1}\{R_n > \gamma\}\right] = \tilde{\mathbb{E}}\left[\sum_{n=1}^{\alpha} \mathbb{1}\{\tilde{R}_n > \gamma\} \cdot L(\tilde{F}_n)\right], \tag{9}$$

where $\tilde{\mathbb{E}}(\cdot)$ is the expectation associated with IS distribution $\tilde{f}_A$ and $\tilde{f}_S$. F_n is the transmission finish time of the n-th flow and $L(t)$ is the likelihood ratio until time t, which will be introduced as follows.

Since IS changes the probability of the original distribution (f_A, f_S), in order to ensure that the expectation of response time R remains the same as that in real-world distribution and conditions, we introduce a weighted term, called *likelihood ratio* [4]. The likelihood ratio L until time t, which considers inter-dependencies between different flows, is given by:

$$L(t) = \frac{\prod_{n=1}^{N_A(t\wedge\tau)} f_A(A_n) \cdot G_A(H^A) \cdot \prod_{n=1}^{N_S(t\wedge\tau)} f_S(S_n)}{\prod_{n=1}^{N_A(t\wedge\tau)} \tilde{f}_A(A_n) \cdot \tilde{G}_A(H^A) \cdot \prod_{n=1}^{N_S(t\wedge\tau)} \tilde{f}_S(S_n)}, \tag{10}$$

where, A_n is the inter-arrival time between the $(n-1)$-th and the n-th arrival of n-th flow, S_n is the service time of n-th flow, $N_A(t)$ is the number of arrival flows until time t, $N_S(t)$ is the number of flows starting transmission until time t, $G_A(t) = e^{-\lambda t}$ is the tail probability of inter-arrival time, τ is the arrival time of first flow, $t \wedge \tau$ is the minimum value between τ and t. Since the flow finish time may not be the same as the arrival time of another flow, we introduce the age term $H^A = T_{\{R_n>\gamma\}} - T_{last}$, where T_{last} is the last arrival time of flows and $T_{\{R_n>\gamma\}}$ is the time that event $\{R_n > \gamma\}$ happens.

Estimation of tail quantile using IS is as follows: Firstly, we calculate the empirical tail probability. Suppose we generate m_{is} regenerative cycles using IS simulation and m_{or} cycles using the original simulation, the empirical tail probability $\hat{P}(R > \gamma)$ of response time γ after using IS is

$$\hat{P}(R > \gamma) = \frac{m_{is}^{-1} \cdot \sum_{i=1}^{m_{is}} \left(\sum_{n=1}^{\tilde{\alpha}_i} \mathbb{1}\{\tilde{R}_{i,n} > \gamma\} \cdot L(\tilde{F}_{i,n})\right)}{m_{or}^{-1} \cdot \sum_{i=1}^{m_{or}} \alpha_i} \tag{11}$$

where we use $\tilde{}$ to denote the samples generated by IS. Secondly, we calculate the empirical tail quantile. Suppose that there is a total of $\beta_{is} = \sum_{i=1}^{m_{is}} \alpha_i$ flows in the m_{is} regenerative cycles, each flow has the corresponding response time $\tilde{R}_n$ and the likelihood ratio L_n. After sorting all β_{is} flows in

Algorithm 1: Adaptive Cross-Entropy Searching

Input: E the number of epoch for update parameter of IS; m_{ce} is the number of; $Q(p)$ is the p-tile we want to estimated;

```
AdaptiveCrossEntropy()
    set μ̃_1 ← μ, σ̃_1 ← σ, γ_max ← 0;
    for i ← 1 to E do
        generate m_ce cycles using IS distribution with μ̃_i,σ̃_i;
        calculate μ̃_{i+1},σ̃_{i+1} based on Equation 16 and 17;
        update γ_max as p-tile estimation using m_ce cycles;
    return μ̃_E, σ̃_E, γ_max
```

ascending order of response time $\tilde{R}_n$, we have the ordered samples $\{(\tilde{R}_1, L_1), \cdots, (\tilde{R}_{i'}, L_{i'}), \cdots, (\tilde{R}_{\beta_{is}}, L_{\beta_{is}})\}$. Combined equations7 and 11, the empirical tail quantile $\hat{Q}(p)$ is given by

$$\hat{Q}(p) = \inf\{\gamma : \hat{P}(R > \gamma) < 1 - p\} = \tilde{R}_{n'} \tag{12}$$

where

$$n' = \min\left\{ n : \sum_{i'=n}^{\beta_{is}} L_{i'} \le (1-p) m_{is} \cdot m_{or}^{-1} \sum_{i=1}^{m_{or}} \alpha_i \right\} \tag{13}$$

4.3.5 Searching Near-optimal IS Distribution

Theoretically, the optimal IS distribution [4] in Equation 9 has the minimum variance and the best performance of IS. However, because of the unknown of $\mathbb{E}[\sum_{n=1}^{\alpha} \mathbb{1}\{R_n > \gamma\}]$, finding the optimal distribution is not practical.

To reduce the variance, we use the cross-entropy method to search for the near-optimal IS distribution [41]. The cross-entropy method uses the KL divergence to measure the distance between two different distributions. Hence, we need to minimize the KL divergence between the original distribution and IS distribution, and choose a distribution by solving the following optimization problem:

$$\arg\max_{\tilde{f}_A, \tilde{f}_S} \mathbb{E}\left[\sum_{n=1}^{\alpha} \mathbb{1}\{R_n \ge \gamma\} \cdot \ln L(t) \right] \tag{14}$$

We assume that the IS distributions belong to the same parametric family as the original distribution and only consider the parameter $\tilde{\mu}, \tilde{\sigma}$ in $\tilde{f}_S$. Using Equation 10 and omitting some constant term, the optimization problem changes to:

$$\arg\max_{\tilde{\mu}, \tilde{\sigma}} \mathbb{E}\left[\sum_{n=1}^{\alpha} \mathbb{1}\{R_n > \gamma\} \left(\sum_{n=1}^{N_S(\alpha \wedge \tau)} \left(\ln \sigma + \frac{(\ln S_n - \tilde{\mu})^2}{2\tilde{\sigma}^2} \right) \right) \right] \tag{15}$$

We can prove Equation 15 has one unique global maximum of function value with respect to $\tilde{\mu}, \tilde{\sigma}$. Hence, the closed-form solutions of $\tilde{\mu}, \tilde{\sigma}$ are:

$$\tilde{\mu} = \frac{\mathbb{E}\left[\left(\sum_{n=1}^{\alpha} \mathbb{1}\{R_n \ge \gamma\} \right) \left(\sum_{n=1}^{N_S(\alpha \wedge \tau)} \ln S_n \right) \right]}{\mathbb{E}\left[\left(\sum_{n=1}^{\alpha} \mathbb{1}\{R_n \ge \gamma\} \right) \cdot N_S(\alpha \wedge \tau) \right]} \tag{16}$$

$$\tilde{\sigma}^2 = \frac{\mathbb{E}\left[\left(\sum_{n=1}^{\alpha} \mathbb{1}\{R_n \ge \gamma\} \right) \left(\sum_{n=1}^{N_S(\alpha \wedge \tau)} (\ln S_n - \tilde{\mu})^2 \right) \right]}{\mathbb{E}\left[\left(\sum_{n=1}^{\alpha} \mathbb{1}\{R_n \ge \gamma\} \right) \cdot N_S(\alpha \wedge \tau) \right]} \tag{17}$$

Algorithm 2: Model-based Tail Quantile Estimation

Input: m_{or} and m_{is} the number of cycles generated by the original distribution and the IS distribution; Original distribution f_A, f_S with parameter λ, μ and σ; IS distribution $\tilde{f}_A$, $\tilde{f}_S$ with parameter $\tilde{\lambda}$, $\tilde{\mu}$ and $\tilde{\sigma}$; threshold response time $\gamma_{max} > Q(p)$; $Q(p)$ is the p-tile we want to estimated; ε is the accuracy of confidence interval; K is length of each section.

```
GenerateISCycle()
    set likelihood ratio L ← 0, isInIS ← True, result list r ← {};
    generate a flow with A_1 and S_1 using IS distribution f̃_A, f̃_S, and call FlowArrival(A_1, S_1) after time A_1;
    wait until all scheduled flows are finished;
    return result list {(R̃_1, L_1), ··· , (R̃_N, L_N)};
FlowArrival(A_n, S_n)
    record last arrival time t_last ← t_now
    update L based on A_n, S_n and Equation 10;
    if t_now < t_stop then
        if isInIS then
            generate a flow with A_{n+1}, S_{n+1} using f̃_A, f̃_S;
        else
            generate a flow with A_{n+1}, S_{n+1} using f_A, f_S;
        call FlowArrival(A_1, S_1) after time A_{n+1};
FlowFinish(R̃_n)
    append response time and likelihood ratio (R̃_n, L_n) to r
    if R̃_n > γ_max then
        update age term H_A ← t_now − t_last in L
        change to original distribution isInIS ← False
ModelBasedEstimation()
    call AdaptiveCrossEntropy() to find IS parameter;
    generate m_or cycles using original distribution;
    generate m_is cycles using GenerateISCycle();
    // batch method
    split m_is cycles into m_b batches, each with length K,
    for i ← 1 to m_b do
        calculate Q̂_i(p) of i-th batch based on Equation 12;
    calculate CI of {Q̂_{m_b}(p)} using model-free algorithm
    return CI
```

Based on the above equation, we can efficiently update the parameters in the distribution of service time. The pseudo-code of the adaptive cross-entropy algorithm is shown in Algorithm 1. We first use the parameter of the original distribution as the starting point for the search. For each epoch, this algorithm generates m_{ce} cycles using IS distribution. Based on these IS cycles, the algorithm can calculate new $\tilde{\mu}_{i+1}$,$\tilde{\sigma}_{i+1}$ based on Equation 16 and 17, and update γ_{max} until the end of E epochs.

5 Implementation

In this section, we present the implementation details of CCEval. We implement CCEval on the NS-3 simulator [38], M3 [28], Parsimon [53], and real-world testbeds.

Input distribution modeling. In the model-based estimation algorithm, CCEval needs the distribution of flow size and inter-arrival time. Hence, we need to find the right parameters such that the difference between the original empirical distribution and the modeled distribution is as small as possible.

For the inter-arrival distribution, we can directly calculate the average flow size based on the empirical CDF and set the parameter λ based on the setting parameter network load u.

Since logarithmic normal distributions are nonlinear and only cumulative distribution functions of flow size are publicly available, CCEval uses the nonlinear least squares method [45] to find the best parameters σ and μ with the minimal difference between two CDF curves.

Model-based tail quantile estimation. The pseudo-code of the model-based tail quantile estimation algorithm is shown in Algorithm 2 based on §4.3.

For each regenerative cycle, we use one type of IS simulation, called the switching change of measure method [4]. Based on this method, each regeneration cycle contains two phases: the IS distribution phase and the original distribution phase. At the start of each cycle, the system is in the IS distribution phase, in which flows are generated based on the IS distributions $\tilde{f}_A$, $\tilde{f}_S$. Once the event $R_N > \gamma_{max}$ occurs, the system switches back to the original distributions f_A and f_S.

CCEval should monitor two types of events in this algorithm: flow arrival and flow completion. In the flow arrival event, the algorithm should record the last arrival time and update the likelihood ratio L. In the flow completion event, the algorithm should record the response time $\tilde{R}_n$ and the corresponding likelihood ratio L_n for each flow. Since the likelihood ratio L multiplies many terms, to improve the numerical stability, we use a logarithmic version of the likelihood ratio.

To further reduce the variance caused by IS, we use the batch method [4]. For m_{is} IS cycles, we split them into m_b batches, and each of them has length K ($m_{is} = K * m_b$). For each batch, the algorithm calculates $\hat{Q}_i(p)$ for the i-th batch based on Equation 12. Finally, this algorithm reuses the model-free algorithm to calculate the confidence interval of quantile.

6 Evaluation

6.1 Experimental Methodology

Infrastructure and testbed. We evaluate CCEval using simulation and a real testbed. Each server has two Intel Xeon Gold 5320 CPUs with 52 physical cores (104 logical cores with hyper-threading) and 256GB DRAM. The servers run Ubuntu 22.04.2. Each server is equipped with a Mellanox ConnectX-3 NIC with 100 Gbps link speed and is connected to two Mellanox SN2700 Ethernet switches. Except in Figure 8 and the testbed experiments of §6.4, all experiments are on NS3 running on a single server.

Target workload and network. To evaluate the generality of CCEval on different flow distributions, we use four publicly available flow distributions: WebSearch, GoogleRPC, AliStorage, and FbHadoop [30, 40, 54]; however, since CCEval has similar results on all of them, we only show results using the WebSearch distribution.

The default topology used in most experiments is a large-scale FatTree topology with $k = 8$, which consists of 320 servers with 100 Gbps NICs and 80 switches with 100 Gbps link bandwidth. In a subset of simulation experiments and our hardware testbed, we also use a small-scale DumbBell topology comprising 10 senders and 10 receivers that share one bottleneck link with 100 Gbps bandwidth.

Target CCA and experimental configuration. To show the generality of CCEval on different CCAs, we evaluate the performance of four widely used CC algorithms: DCTCP [3], DCQCN [54], TIMELY [35], HPCC [30]. For each CCA, we use the default hyperparameters in the original paper.

As the default setting, we set the duration of each trial as 0.1 seconds and network load as 30%. We report the flow completion time with different quantiles (90th, 95th, 99th, and 99.9th) of each experiment.

Hyper-parameters of CCEval. Unless otherwise specified, we set the confidence level, $(1-\rho)$, as 95% with a target confidence interval of 1% relative margin of error, *i.e.*, $1\% * observed\ mean$. For the model-free algorithm, we use $n = 20$ in the exploratory stage. For the adaptive cross-entropy algorithm, we use $E = 10$ epochs and each epoch generates $m_{ce} = 10$ IS cycles to update the parameters $\tilde{\mu}$, $\tilde{\sigma}$ of the IS distribution. We use $m_{or} = 100$ to estimate the length of each cycle in the original distribution, $K = 40$ for the batch method in the model-based algorithm.

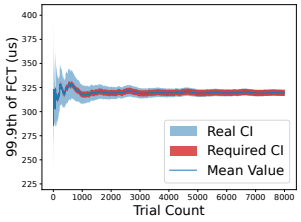


Figure 7: Real CI converges to the required CI after more trials, using TIMELY on a FatTree.

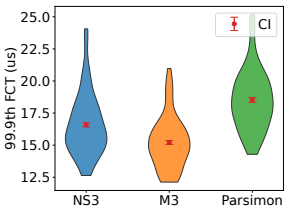
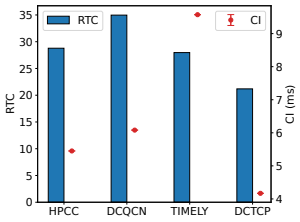


Figure 8: CCEval calculates CI for NS3, M3 and Parsimon with 1% error and 95% confidence.

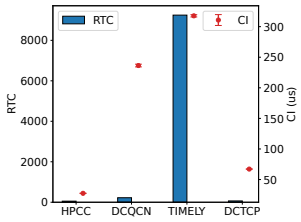
6.2 CCEval Evaluates Performance Metrics Accurately and Confidently

We show that CCEval can calculate confidence intervals accurately for different quantiles of different CCAs. We use the model-free estimation algorithm to calculate the confidence interval given a target of 1% relative margin of error and report the real trial count when we reach convergence.

Figure 7 compares—for WebSearch traffic and TIMELY on a FatTree—the target (*i.e.*Required) CI versus the current

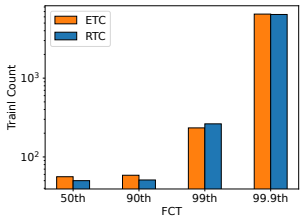


(a) 50th FCT of large flows

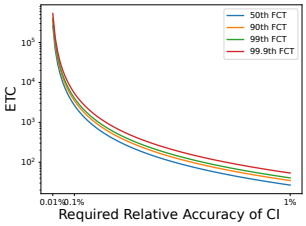


(b) 99.9th FCT of small flows

Figure 9: Final confidence interval (CI) and real trial count (RTC) of different CCAs, using model-free estimation algorithm with 1% error and 95% confidence level on WebSearch and FatTree.

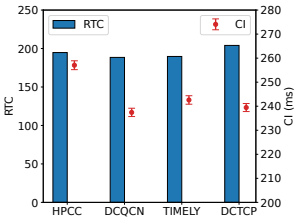


(a) ETC and RTC are similar.

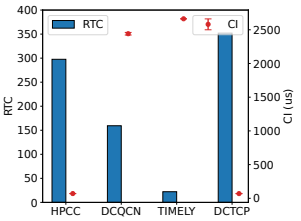


(b) ETC increases with smaller error.

Figure 10: Compared to the real trial count (RTC), CCEval can forecast the estimated trial count (ETC) accurately achieving 1% relative error. RTC increases significantly with smaller relative error.



(a) 50th FCT of large flows



(b) 99.9th FCT of small flows

Figure 11: Confidence interval (CI) and real trial count (RTC) of different settings, using model-free estimation algorithm with 1% error and 95% confidence level on WebSearch and Dumbbell.

(*i.e.*Real) CI. In the initial few trials, the difference between the real CI and the required CI is large; however, the real CI quickly converges to satisfy the required 1% margin of error over the course of a few thousand trials.

Our tool can also be applied to other network simulators, and we demonstrate that in Figure 8 by applying CCEval to M3 [28] and Parsimon [53]. Aside from demonstrating the generality of CCEval, the figure also illustrates an important point: because M3 and Parsimon use machine learning methods to approximate the simulation results, their final estimates of the mean 99th quantile FCT differ from those of NS3 (by 10–20%). CCEval cannot compensate for potential inaccuracies in the underlying experiments; it instead focuses on the effect of varying workload conditions on the mean of the experimental results (in this case, 99th quantile FCTs).

In Figure 9, for the median FCT of large flows and the tail quantile of small flows in the four CCAs, we report the final mean/CI as well as the number of trials CCEval needed to run to get to 1% relative margin of error with 95% confidence. In Figure 9b, we see that TIMELY needs many more trials than other CCAs, since, as we observed in Figure 3b, its performance is variable, likely due to its sensitivity to fine-grained delays. A detailed discussion of this phenomenon is beyond the scope of our paper, but it illustrates the types of insights we can gain from careful analysis of multi-trial evaluations.

6.3 CCEval Forecasts Trial Count Accurately

We compare the estimated trial count (ETC) based on Equation 3 and the real trial count (RTC) recorded when the model-free algorithm converges to a target CI, given a target CI.

As shown in Figure 10a, CCEval can forecast the RTC accurately, with forecasting errors ranging from 1% to 10%. Errors come mainly from the intrinsic randomness of sampling from the flow distribution. One way to reduce these errors is to increase the number of trials in the exploratory stage.

Figure 10b shows how the ETC changes for different relative margins of error. Essentially, ETC increases quadratically as you try to increase accuracy and reduce the CI. If we want to achieve an accuracy of the CI as small as 0.01%, we potentially need billions of trials!

Taking a step back, our results indicate a few things: *(1) More trials are needed to converge to accurate evaluation results.* As mentioned in §2.1, current evaluation workflows use a very limited number of trials (≤ 10). However, Figure 10 indicates that we need at least 20 trials to converge to 1% relative margin of error at a 95% confidence level. *(2) RTC increases significantly as we move further toward the tail.* For example, in Figure 10a, the 50th FCT of TIMELY only needs 27 trials to converge, but the 99.9th FCT needs 9246 trials (340×). This phenomenon comes from the large variance of 99.9th FCT due to the small probability of occurrence.

6.4 CCEval Generalizes to Different Settings

Different network topologies. To show the generality of CCEval, we extend simulation experiments to a small Dumbbell topology. We use the same WebSearch distribution and trial duration as those in the previous section (0.1 s). As shown in Figure 11, CCEval can also be applied directly to this topology. In most cases, the RTC in the DumbBell experiments is larger than that of the FatTree, as all flows contend on the bottleneck link. The exception is TIMELY in the case of small flows. We hypothesize that TIMELY's delay-based congestion model may be more accurate if there is only a single shared source of delay among all flows.

Hardware testbed. We also evaluate CCEval on a hardware testbed. The testbed uses the same DumbBell topology and experimental settings as Figure 11, except that the testbed only evaluates DCTCP as it is the only CCA currently supported by Linux and our network infrastructure.

Figure 12 reports the RTC and mean/CI of different FCT quantiles. We observe a similar effect of the tail needing higher RTCs to converge. Still, we find that the hardware

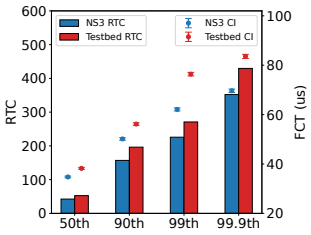
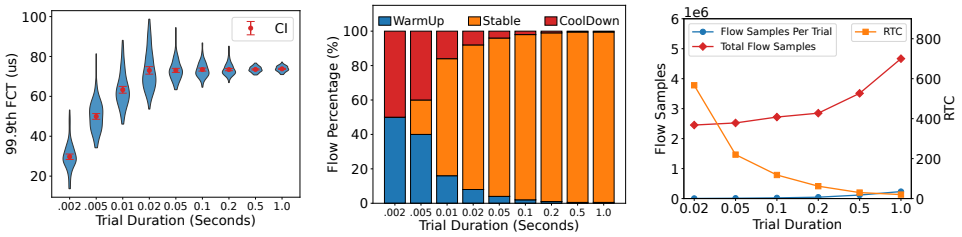


Figure 12: Confidence interval (CI) and real trial count (RTC) on NS3 and real-world testbed.



(a) CI and original distribution (b) Flow percentage in three phases. (c) Total flow samples VS durations

Figure 13: We vary the trial duration from 0.002 s to 1.0 s and show impact on CI, RTC, flow percentage in three phases (WarmUp, Stable, CoolDown), and total flow samples versus duration, using DCTCP and the model-free estimation algorithm with 1% margin of error and 95% confidence level on WebSearch and FatTree.

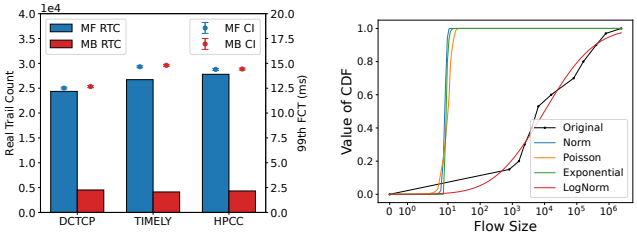
testbed needs more trials (>20% more) to converge to the target CI of 1% relative error, which implies that, perhaps unsurprisingly, the results of real-world experiments have higher variability. This observed higher variability may come from the intrinsic variability of hardware or the OS [34], which is not captured in simulations.

Different traffic durations. Next, we vary the traffic duration to show how this hyperparameter influences the evaluation and CI. In Figure 13a, we report the CI and original distribution (100 trials) of 99.9th FCT for small flows (WebSearch with DCTCP on a FatTree). We note that durations beyond 1 s shows similar results, but is progressively more expensive to simulate to CI convergence. We can infer a couple of important insights from this data.

(1) Due to a large proportion of non-representative flows, very-short-duration experiments provide an incorrect estimation of performance. If researchers set the duration as low as 0.002s, they may conclude that the mean 99.9th FCT is 29.72 us. However, running for longer, we can see that for sufficiently long experiments, the mean 99.9th FCT converges closer to 73.42 us. As shown in Figure 13b, we find that the issues come from the fact that very short trial durations are dominated by the ramp-up/down behavior of flows, rather than their steady-state behavior. Specifically, each experiment contains three phases: a warm-up phase (active flows increase from zero), a stable phase (flows generate and exit normally), and a cool-down phase (active flows decrease to zero). The warm-up and cool-down phases are unstable, and flows in these phases are non-representative, and thus underestimates the impact of network congestion. For the 0.002s duration, for instance, >90% of the experiment is warm-up or cool-down.

(2) Very-long-duration experiments need more total flow samples and longer runtime to converge to the required relative accuracy. Figure 13c reports the flow samples per trial (FSPT), the RTC, and the total number of flows needed to converge ($RTC * FSPT$). FSPT increases linearly with longer trial duration, but the RTC decreases more slowly. Hence, the total number of flows to converge goes up when the experimental setting uses a longer trial duration.

Based on these findings, trial duration should be carefully selected. The sweet spot is the minimal duration that achieves stable estimation, *e.g.*, 0.02 s in Figure 13. Luckily, this bias toward shorter trials lends itself to our model-based approach and also to efficient parallel execution.



(a) Model-free VS model-based (b) Curve fitting of websearch

Figure 14: Compared to model-free algorithm, model-based estimation algorithm can reduce real trial count significantly while achieving the same 1% error and 95% confidence of CI.

6.5 CCEval Reduces Runtime Significantly

To achieve accurate and confident estimation of performance metrics, researchers need long experimental runtime due to large number of trials. CCEval can use the model-based tail quantile estimation algorithm to reduce real trial count and experimental runtime significantly. We use a Dumbbell topology with trial duration (0.02s) and relative error 1%, and report the 99th FCT of all flows. As shown in Figure 14a, we report the real trial count (RTC) and confidence interval (CI) of model-free and model-related algorithms to achieve the same relative accuracy. Compared to the model-free algorithm, the model-based algorithm can reduce real trial count by 75%, 80% and 80% for DCTCP, TIMELY, and HPCC, respectively, while achieving the same CI.

The significant reduction comes from two aspects. First, we model the probability distribution of flow size accurately. As shown in Figure 14b, we use four different widely used probability distributions: normal, log-normal, Poisson, and exponential. Only the lognormal distribution has the minimum fitting error with the empirical CDF of the real traffic distribution. Second, we use importance sampling to increase the probability of observing the tail quantile. For example, for flows with a response time greater than a given value ($R_n > 11944701$ ns), the model-based algorithm increases the probability 3.5 times, from 1.24% to 4.32%.

6.6 CCEval Has Low System Overhead

CCEval incurs very low system overhead. For the model-free algorithm, CCEval only needs to calculate the empirical mean and variance. All experimental results in the exploratory stage can be used in the production stage, without being wasted. As a result, the model-free estimation algorithm introduces less than 0.1% runtime overhead.

For the model-based algorithm, CCEval has three types of overheads. First, it uses the nonlinear least squares method to model the probability distribution of flow size, which can be done offline. Second, extra trials are needed to find the near-optimal IS distribution. Third, it generates m_{or} trials to calculate the required term in Equation 13. Only the second and third types of overheads are changed for different experimental settings and contribute to runtime overhead. Overall, this algorithm introduces 2.5% runtime overhead, which is affordable compared to an 80% reduction of trial count.

7 Discussion

What are the assumptions of CCEval? Is CCEval specific to congestion control? CCEval proposes two estimation algorithms, a model-free algorithm and a model-based algorithm. The model-free algorithm only relies on the independent and identically distributed (i.i.d.) assumption[1] of different trials; hence, it is not specific to congestion control and can apply to general performance metrics in other areas [34, 47]. The model-based algorithm relies on the modeling of the input and output distributions of CCA experiments and uses an importance-sampling method that leverages domain-specific knowledge of congestion control. As a result, the model-based algorithm is specific to congestion control and tail quantile estimation.

How does CCEval deal with flow traces? As shown in Figure 1, CCEval targets flow-distribution workloads as CCEval needs to repeatedly sample traffic workload from the flow distribution and further needs to model the distribution using a logarithmic normal distribution. If researchers only have flow traces, they need to extract important characteristics of the trace data, such as flow distribution, and generate synthetic traffic to match these characteristics [54].

How can CCEval model traffic burstiness? Traffic in DCNs is infamously bursty [51], greatly impacting tail latency. CCEval follows the experimental methodology used in current CCA evaluations [5, 6, 30]. In the model-based tail quantile estimation algorithm, CCEval uses an exponential distribution with rate λ to model the inter-arrival time of flows, leading to a Poisson process of flow arrivals with the same rate, λ. By changing the traffic load percentage, CCEval can change the degree of burstiness in flow arrivals. In principle, CCEval may also be applied to other traffic patterns, *e.g.*, long-tailed distributions like lognormal or Pareto [7]; however, we leave this as future work. Regardless, the model-free algorithm is universally applicable to these alternative traffic patterns.

[1] In practice, workloads may exhibit correlations or cannot apply the i.i.d. assumption and CLT. We leave relaxing these assumptions as future work.

8 Related Work

CCA and CCA evaluation. Numerous congestion control algorithms have been proposed since the occurrence of computer networks [26]. Based on the congestion signal used, CCAs can be classified as loss-based [14, 26], delay-based [35], ECN-based [3], and INT-based [30]. Researchers find CCA performance varies across different network scenarios and build benchmarking tools [2, 49] based on physical testbeds, simulators [15, 25, 38], or emulators [31] to conduct experiments [17, 18, 27, 37]. However, none of them focus on the variability and confidence of multi-trial evaluation.

Queuing systems. Some works on queuing systems are close to CCEval. Most of the works focus on single-switch systems [9, 19, 33]. They either need to model the behavior of a single switch or strongly assume queuing systems will enter a steady state. Since real networks contain many switches and dynamically change due to different flows and CCAs, they cannot be applied to real networks with complex CCAs.

Approximation of CCA trials. Some works (Parsimon [53], m3 [28], and MimicNet [52]) have been proposed for fast and accurate estimation of the tail latency. They still follow the current evaluation workflow and can speed up each trial of CCA experiments compared to NS-3. As shown in §6.2, CCEval is largely orthogonal, but the runtime of CCEval could be further reduced by using these systems.

9 Conclusion

In this paper, we present CCEval. CCEval proposes using confidence intervals to quantify and improve the accuracy and confidence of performance results, and designs a model-free estimation algorithm to calculate the confidence interval. We further design a model-based tail quantile estimation algorithm, which can significantly reduce trial count without loss of accuracy and confidence. Our extensive experiments show that CCEval produces an accurate and confident estimation of performance metrics for four mainstream CCAs on typical DCN topologies and flow distributions, and reduces trial count by 75%∼80% for tail quantile estimation.

Acknowledgement

We thank our shepherd Prof. Soudeh Ghorbani and the anonymous NSDI reviewers for their constructive comments. Kaihui Gao and Li Chen are the corresponding authors. This work was supported by the National Key R&D Program of China under grant 2024YFA1014203, the Beijing Outstanding Young Scientist Program (No. JWZQ20240101008), and Zhongguancun Laboratory.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, pages 265–283. USENIX Association, 2016.

[2] Soheil Abbasloo. Internet congestion control benchmarking. *CoRR*, abs/2307.10054, 2023.

[3] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 63–74. ACM, 2010.

[4] Søren Asmussen and Peter W. Glynn. *Stochastic simulation - algorithms and analysis*, volume 57 of *Stochastic modeling and applied probability*. Springer, 2007.

[5] Songyuan Bai, Hao Zheng, Chen Tian, Xiaoliang Wang, Chang Liu, Xin Jin, Fu Xiao, Qiao Xiang, Wanchun Dou, and Guihai Chen. Unison: A Parallel-Efficient and User-Transparent Network Simulation Kernel. In *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*, pages 115–131. ACM, 2024.

[6] Wei Bai, Li Chen, Kai Chen, and Haitao Wu. Enabling ECN in Multi-Service Multi-Queue Data Centers. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, pages 537–549. USENIX Association, 2016.

[7] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, pages 267–280. ACM, 2010.

[8] Mark Berman, Jeffrey S. Chase, Lawrence H. Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. GENI: A federated testbed for innovative network experiments. *Comput. Networks*, 61:5–23, 2014.

[9] Jose H. Blanchet, Peter W. Glynn, and Jingchen Liu. Fluid heuristics, lyapunov bounds and efficient importance sampling for a heavy-tailed G/G/1 queue. *Queueing Syst. Theory Appl.*, 57(2-3):99–113, 2007.

[10] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. Computer and communication sciences. CRC Press, 2010.

[11] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: congestion-based congestion control. *Commun. ACM*, 60(2):58–66, 2017.

[12] Michael Allen Crane and Austin Joseph Lemoine. *An introduction to the regenerative method for simulation analysis*. Springer, 1977.

[13] Edgar C. Fieller. Some problems in interval estimation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 16(2):175–185, 1954.

[14] Sally Floyd. Highspeed TCP for large congestion windows. *RFC*, 3649:1–34, 2003.

[15] Kaihui Gao, Li Chen, Dan Li, Vincent Liu, Xizheng Wang, Ran Zhang, and Lu Lu. DONS: Fast and Affordable Discrete Event Network Simulation with Automatic Parallelization. In *Proceedings of the ACM SIGCOMM 2023 Conference, ACM SIGCOMM 2023, New York, NY, USA, 10-14 September 2023*, pages 167–181. ACM, 2023.

[16] Yixiao Gao, Yuchen Yang, Chen Tian, Jiaqi Zheng, Bing Mao, and Guihai Chen. DCQCN+: Taming Large-Scale Incast Congestion in RDMA over Ethernet Networks. In *2018 IEEE 26th International Conference on Network Protocols, ICNP 2018, Cambridge, UK, September 25-27, 2018*, pages 110–120. IEEE Computer Society, 2018.

[17] Luca Giacomoni and George Parisis. Reinforcement Learning-based Congestion Control: A Systematic Evaluation of Fairness, Efficiency and Responsiveness. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications, Vancouver, BC, Canada, May 20-23, 2024*, pages 1451–1460. IEEE, 2024.

[18] Luigi Alfredo Grieco and Saverio Mascolo. Performance evaluation and comparison of Westwood+, New Reno, and vegas TCP congestion control. *Comput. Commun. Rev.*, 34(2):25–38, 2004.

[19] Jin Guang, Guiyu Hong, Xinyun Chen, Xi Peng, Li Chen, Bo Bai, and Gong Zhang. Tail Quantile Estimation for Non-Preemptive Priority Queues. In *Winter Simulation Conference, WSC 2022, Singapore, December 11-14, 2022*, pages 85–96. IEEE, 2022.

[20] Fei Gui, Kaihui Gao, Li Chen, Dan Li, Vincent Liu, Ran Zhang, Hongbing Yang, and Dian Xiong. Accelerating Design Space Exploration for LLM Training Systems with Multi-experiment Parallel Simulation. In *22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 2025, Philadelphia, PA, USA, April 28-30, 2025*, pages 473–488. USENIX Association, 2025.

[21] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible network experiments using container-based emulation. In *Conference on emerging Networking Experiments and Technologies, CoNEXT '12, Nice, France - December 10 - 13, 2012*, pages 253–264. ACM, 2012.

[22] Stephen Hemminger. Network emulation with NetEm. In *Linux conf au*, volume 5, page 2005, 2005.

[23] Jerry L. Hintze and Ray D. Nelson. Violin plots: a box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

[24] Torsten Hoefler and Roberto Belli. Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, November 15-20, 2015*, pages 73:1–73:12. ACM, 2015.

[25] Teerawat Issariyakul, Ekram Hossain, Teerawat Issariyakul, and Ekram Hossain. *Introduction to network simulator 2 (NS2)*. Springer, 2009.

[26] Van Jacobson. Congestion avoidance and control. *Comput. Commun. Rev.*, 25(1):157–187, 1995.

[27] Huiling Jiang, Qing Li, Yong Jiang, Gengbiao Shen, Richard O. Sinnott, Chen Tian, and Mingwei Xu. When machine learning meets congestion control: A survey and comparison. *Comput. Networks*, 192:108033, 2021.

[28] Chenning Li, Arash Nasr-Esfahany, Kevin Zhao, Kimia Noorbakhsh, Prateesh Goyal, Mohammad Alizadeh, and Thomas E. Anderson. m3: Accurate Flow-Level Performance Estimation using Machine Learning. In *Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM 2024, Sydney, NSW, Australia, August 4-8, 2024*, pages 813–827. ACM, 2024.

[29] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, 2020.

[30] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. HPCC: high precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, pages 44–58. ACM, 2019.

[31] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P. Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. CrystalNet: Faithfully Emulating Large Production Networks. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 599–613. ACM, 2017.

[32] Tianfeng Liu, Yangrui Chen, Dan Li, Chuan Wu, Yibo Zhu, Jun He, Yanghua Peng, Hongzheng Chen, Hongzhi Chen, and Chuanxiong Guo. BGL: GPU-Efficient GNN Training by Optimizing Graph Data I/O and Preprocessing. In *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*, pages 103–118. USENIX Association, 2023.

[33] Michel Mandjes and Bert Zwart. Large deviations of sojourn times in processor sharing queues. *Queueing Syst. Theory Appl.*, 52(4):237–250, 2006.

[34] Aleksander Maricq, Dmitry Duplyakin, Ivo Jimenez, Carlos Maltzahn, Ryan Stutsman, Robert Ricci, and Ana Klimovic. Taming Performance Variability. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*, pages 409–425. USENIX Association, 2018.

[35] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily R. Blem, Hassan M. G. Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, pages 537–550. ACM, 2015.

[36] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using megatron-lm. In Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin, editors, *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2021, St. Louis, Missouri, USA, November 14-19, 2021*, page 58. ACM, 2021.

[37] Truc Anh N. Nguyen, Siddharth Gangadhar, and James P. G. Sterbenz. Performance evaluation of TCP congestion control algorithms in data center networks. In *Proceedings of the 11th International Conference on Future Internet Technologies, CFI 2016, Nanjing, China, June 15-17, 2016*, pages 21–28. ACM, 2016.

[38] George F. Riley and Thomas R. Henderson. The ns-3 network simulator. *Modeling and tools for network simulation*, pages 15–34, 2010.

[39] Vijay K. Rohatgi and A.K. Md. Ehsanes Saleh. *An introduction to probability and statistics*. John Wiley & Sons, 2015.

[40] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, pages 123–137. ACM, 2015.

[41] Reuven Y. Rubinstein and Dirk P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

[42] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

[43] Walter L. Smith. Regenerative stochastic processes. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 232(1188):6–31, 1955.

[44] Vaidyanathan Sundarapandian. *Probability, statistics and queuing theory*. PHI Learning Pvt. Ltd., 2009.

[45] Peter Teunissen. Nonlinear least squares. 1990.

[46] Surya T. Tokdar and Robert E. Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

[47] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan S. Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. Is Big Data Performance Reproducible in Modern Cloud Networks? In *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 513–527. USENIX Association, 2020.

[48] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Dan Li, Li Chen, Heyang Zhou, Linkang Zheng, Sen Zhang, Yikai Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, Kunling He, Jiaqi Gao, Ennan Zhai, Dennis Cai, and Binzhang Fu. SimAI: Unifying Architecture Design and Performance Tuning for Large-Scale Large Language Model Training with Scalability and Precision. In *22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 2025, Philadelphia, PA, USA, April 28-30, 2025*, pages 541–558. USENIX Association, 2025.

[49] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Alexander Levis, and Keith Winstein. Pantheon: the training ground for internet congestion-control research. In *Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*, pages 731–743. USENIX Association, 2018.

[50] Qingqing Yang, Xi Peng, Li Chen, Libin Liu, Jingze Zhang, Hong Xu, Baochun Li, and Gong Zhang. DeepQueueNet: towards scalable and generalized network performance estimation with packet-level visibility. In Fernando Kuipers and Ariel Orda, editors, *SIGCOMM '22: ACM SIGCOMM 2022 Conference, Amsterdam, The Netherlands, August 22 - 26, 2022*, pages 441–457. ACM, 2022.

[51] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. High-resolution measurement of data center microbursts. In Steve Uhlig and Olaf Maennel, editors, *Proceedings of the 2017 Internet Measurement Conference, IMC 2017, London, United Kingdom, November 1-3, 2017*, pages 78–85. ACM, 2017.

[52] Qizhen Zhang, Kelvin K. W. Ng, Charles W. Kazer, Shen Yan, João Sedoc, and Vincent Liu. MimicNet: fast performance estimates for data center networks with machine learning. In *ACM SIGCOMM 2021 Conference, Virtual Event, USA, August 23-27, 2021*, pages 287–304. ACM, 2021.

[53] Kevin Zhao, Prateesh Goyal, Mohammad Alizadeh, and Thomas E. Anderson. Scalable tail latency estimation for data center networks. In *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*, pages 685–702. USENIX Association, 2023.

[54] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale RDMA deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, pages 523–536. ACM, 2015.