Modeling and Optimization of Straggling Mappers

F. Farhat, D. Z. Tootaghaj, A. Sivasubramaniam, M. Kandemir, C. R. Das

Computer Science and Engineering, The Pennsylvania State University, University Park, USA
{fuf111,dxz149,anand,kandemir,das}@cse.psu.edu

ABSTRACT

MapReduce framework is widely used to parallelize batch jobs since it exploits a high degree of multi-tasking to process them. However, it has been observed that when the number of mappers increases, the map phase can take much longer than expected. This paper analytically shows that stochastic behavior of mapper nodes has a negative effect on the completion time of a MapReduce job, and continuously increasing the number of mappers without accurate scheduling can degrade the overall performance. We analytically capture the effects of stragglers (delayed mappers) on the performance. Based on an observed delayed exponential distribution (DED) of the response time of mappers, we then model the map phase by means of hardware, system, and application parameters. Mean sojourn time (MST), the time needed to sync the completed map tasks at one reducer, is mathematically formulated. Following that, we optimize MST by finding the task inter-arrival time to each mapper node. The optimal mapping problem leads to an equilibrium property investigated for different types of inter-arrival and service time distributions in a heterogeneous datacenter (i.e., a datacenter with different types of nodes). Our experimental results show the performance and important parameters of the different types of schedulers targeting MapReduce applications. We also show that, in the case of mixed deterministic and stochastic schedulers, there is an optimal scheduler that can always achieve the lowest MST.

Keywords

Queuing Theory, Performance, Modeling, Optimization, MapReduce.

1. INTRODUCTION

MapReduce has become a popular paradigm for structuring large scale parallel computations in datacenters. By decomposing a given computation into (one or more) Map and Reduce phases, the work within each phase can be accomplished in parallel without worrying about data dependencies, and it is only at the boundaries between these phases where one needs to worry about issues such as data availability and dependency enforcement. At the same time, with the possibility of elastically creating tasks of different sizes within each phase, these computations can adjust themselves to the dynamic capacities available in the datacenter. There has been a lot of prior work in the past decade to leverage this paradigm for different applications [1,2,3], as well as in the systems substrate needed to efficiently support their execution at runtime [4,5,6].

While each phase is highly parallel, the inefficiencies in MapReduce execution manifest at the boundaries between the phases as data exchanges and synchronization stalls, which ensure completion of the prior phases. One of these inefficiencies is commonly referred to as the *straggler problem* of mappers, where a reduce phase has to wait until all mappers have completed their work [4]. Even if there is one such straggler, the entire computation is consequently slowed down. Prior work [7,8,9,10] has identified several reasons for such stragglers including load imbalance, scheduling inefficiencies, data locality, communication overheads, etc. There have also been efforts looking to address one or more of these concerns to mitigate the straggler problem [7,8,11,12,13]. While all these prior efforts are important, and useful to address this problem, we believe that a rigorous set of analytical tools is needed in order to: (i) understand the consequences of stragglers on the performance slowdown in MapReduce execution; (ii) be able to quantify this slowdown as a function of different hardware (processing speed, communication bandwidth, etc.), system (scheduling policy, task to node assignment, data distribution, etc.), and application (data size, computation needs, etc.) parameters; (iii) study the impact of different scaling strategies (number of processing nodes, the computation to communication and data bandwidths, tasks per node, etc.) on this slowdown; (iv) undertake "what-if" studies for different alternatives (alternate scheduling policies, task assignments to nodes, etc.) beyond what is available to experiment with on the actual platform/system; and (v) use such capabilities for a wide range of optimizations: they could include determining resources (nodes, their memory capacities, etc.) to provision for the MapReduce jobs, the number of tasks to create and even adjust dynamically, the assignment of these tasks to different kinds of nodes (since datacenters could have heterogeneous servers available at a given time), adjusting the scheduling policies, running redundant versions of the tasks based on the trade-offs between estimated wait times and additional resources mandated, executing a MapReduce computation under a budgetary (performance, power, cost) constraint, etc.

To our knowledge, there are no rigorous analysis tools available today with these capabilities for modeling and understanding the straggler problem in MapReduce for the purposes listed above. This paper intends to fill this critical gap by presenting an *analytical model* for capturing the waiting time at the end of the Map phase due to any straggling mappers. We also demonstrate the benefits of having such a tool with a few case studies. Specifically, this paper makes the following **contributions** towards presenting and exploiting an analytical model for the stragglers in MapReduce computations:

- We demonstrate that a *delayed exponential distribution* can be used to capture the service time of the map tasks at a given node. We then show that with such service times, the aggregate completion time of mapper tasks across all the nodes of the cluster also follows a delayed exponential distribution. This is validated (less than 5% least square error) against a spectrum of mapper completion times of 10 production workloads presented in prior research.
- With this result, we develop a *closed-form queuing model* of the time expended (the Mean Sojourn Time) before a reducer node can begin its part of the computation, i.e., waiting time for all mappers to finish. Parameterized by the inter-arrival time of the mappers, the delayed exponential service times, and the number of nodes, this model helps conveniently study the impact of different parameters--whether job characteristics, hardware capabilities or system operation--on the delays before a reducer can start.
- This model can be used for a variety of purposes as explained above. In this paper, we specifically illustrate two use cases. First, we show how the model can be used to *schedule map tasks* on different (possibly heterogeneous) nodes of a datacenter to reduce the Mean Sojourn Time (MST). This is demonstrated to be much more effective than the JobTracker scheduling of the current Hadoop distribution [5]. Second, we show that increasing the nodes assigned to map tasks is not always helpful, since the variance of completion times can increase the Mean Sojourn Time. There is, indeed, a critical number of nodes that should be assigned to map tasks, and we illustrate how our model can be used to determine it.

2. BACKGROUND AND RELATED WORK

2.1 MapReduce Framework

The MapReduce framework [4] is a programming paradigm that can be used to execute data-intensive jobs. This framework can be applied to a large class of algorithms, known as Map-Reduce Class (MRC) [14], with high levels of parallelism and low job completion times. The open-source Hadoop/MapReduce framework is a fault-tolerant scalable implementation built upon Hadoop file system (HDFS) [5].

Figure 1 shows a high-level view of the MapReduce framework. A job arrives with mean rate λ, and is partitioned into 'map' tasks. More specifically, the JobTracker module in Hadoop assigns map/reduce tasks to TaskTracker nodes. Each map task tracker node (mapper) has threads to perform the map tasks. Once the map tasks are completed, a set of intermediate key/value pairs is generated and passed to the associated reducer node in the shuffling stage. Each reducer node may receive values with the same intermediate key assigned to that node. Each reducer node employs reduce task trackers to compute and merge the received intermediate values. After the reduce phase, the final values are merged into the HDFS storage.

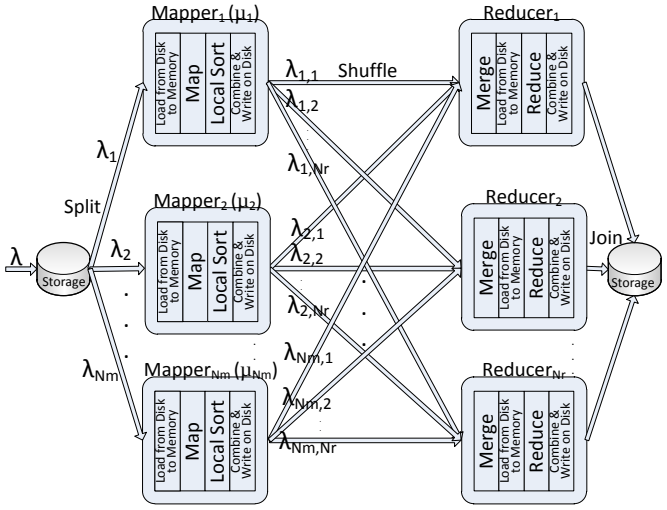


Figure 1. MapReduce framework.

For example, in a word-count application (where the goal is finding the frequency of the words in a huge document), a mapper node may count the frequency of each word in its received text in the map phase, and then send (word, frequency) tuples for assigned reducers in the shuffle phase. It may send words that begin with letter "A" to the first reducer node, words that begin with letters "B, C or D" (having balanced number of words for each) to the second reducer node, etc. It is important to note that the map and shuffle phases may overlap. Each reducer node calculates the total frequency of its words by summing the corresponding frequencies of each word in reduce phase.

For a reducer to start its execution, all mappers that have data to send to that reducer should finish sending. During execution, one may observe various imbalances across mappers, due to resource contention in a mapper node, unbalanced jobs scheduled on mapper nodes or heterogeneity across computational resources [8]. Clearly, the start time of a reduce job is dictated by slowest mapper, that is, the slowest mapper determines how soon a reduce task can start its execution. One of the major reasons for excessively long execution latencies of MapReduce jobs is stragglers, i.e., some cluster nodes that complete their assigned tasks in a longer time than usual. Node hardware failure, resource contention of tasks running on a node, and limited resource availability on a node can make it a straggler.

2.2 Related Work

The mathematical modeling of MapReduce framework has been investigated in several studies [15,16,17,18]. The main difference between these models and our work is that the prior models are not sufficiently rigorous in handing the stragglers problem. Also, most of the published studies assume *deterministic* execution times for mappers and reducers. Li et al. [15] introduce an analytical model for I/O cost, number of I/O requests, and startup cost in Hadoop. They also propose a hash-based mechanism to allow incremental processing and in-memory processing of frequent keys investigating the best merge factor for jobs larger than memory size. Ananthanarayanan et al. [11] show that stragglers can slow down small jobs by as much as 47%. They propose a system called Dolly for cloning small jobs. They also claim that a delay assignment can improve resource contention initiated by cloning. However, this method does not work for stragglers of large tasks. In [16], Ahmad et al. try to find an analytical model of MapReduce to minimize the execution time and find optimal map granularity. In comparison, Karloff et al. [17] present a performance model to estimate the cost of map and reduce functions in MapReduce. However, the model assumes that all mappers finish at the same time and they do not consider the stochastic behavior of execution times of mappers and reducers. Krevat et al. [18] proposed a simple analytical model for MapReduce and compared the performance results of MapReduce with other similar frameworks in some good conditions.

LATE [7] tries to optimize MapReduce job performance in a heterogeneous cluster by restarting slow tasks in fast mapper nodes. Tarazu [12] addresses the poor performance of MapReduce in heterogeneous clusters and shows that the traffic contention between remote tasks is the main problem in heterogeneous clusters. Motivated by this, they then propose a communication-aware and dynamic load balancing technique to reduce the network traffic contention between remote tasks and the shuffling stage. SkewTune [19] interactively manages the skew in non-uniform input data at runtime. It finds an idle node in the cluster and assigns a slow task to that node. Ananthanarayanan et al. [8] discuss the main causes of the outliers (stragglers) and they propose Mantri to restart or duplicate the task at the beginning of its lifetime. Scarlett [13] replicates popular blocks across different memory components to reduce interference with running jobs.

Also, detailed analyses of various workloads [9] such as OpenCloud, M45 and WebMining, show stragglers runtime to median task duration has a nearly delayed exponential distribution. Similarly, Chen et al. [10] evaluate the task lengths for Cloudera and Facebook workloads and reach similar conclusions. Tan et al. [20] propose a coupling scheduler in MapReduce based on an analytical approach and also model FIFO and fair scheduler with an index range for delay distribution tail. They extended their work [21] by upgrading reducerers as multi-server queue. Lin et al. [22] try to address the challenge of overlapping map and shuffle in MapReduce. They demonstrate that the optimal solution is NP-hard in the offline mode, and they suggest MaxSRPT and SplitSRPT schedulers for online mode, reaching optimal scheduling. Condie et al. [23] change the MapRedduce framework to support pipelining between the map, shuffle and reduce phases. There are some compiler-based architecture [24,25] for SQL-like queries in MapReduce framework to setup and speedup the execution of computation on large data sets. They work on a computational DAG (directed acyclic graph) representation of large queries and need multiple rounds of MapReduce for their optimizations to be effective. A deadline-aware scheduler has been proposed [26] to practically address scheduling of deadline-sensitive jobs in a satisfactory range.

2.3 Delayed Exponential Distribution

We start by giving the definition of *Delayed Exponential Distribution (DED)* since it is heavily used in our model.

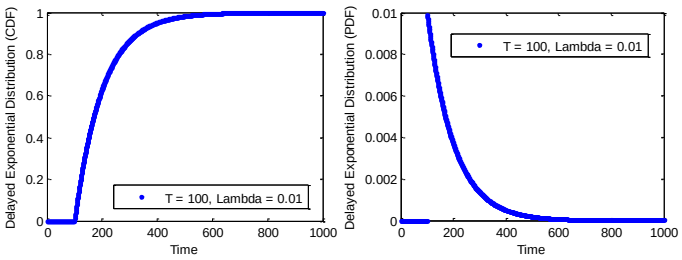
Definition 1: Given arrival rate λ and offset time T, $F(t)$ is the cumulative distribution function (CDF) and $f(t)$ is the probability density function (PDF) of a DED:

$$F(t) = \begin{cases} 0 & ;t < T \\ 1 - e^{-\lambda(t-T)} & ;t \geq T \end{cases} = \left(1 - e^{-\lambda(t-T)}\right)U(t-T), \quad (1)$$

$$f(t) = \begin{cases} 0 & ;t < T \\ \lambda e^{-\lambda(t-T)} & ;t \geq T \end{cases} = \lambda e^{-\lambda(t-T)}U(t-T), \quad (2)$$

where $U(t)$ is unit step function. Figure 2.a and 2.b plot, respectively, the CDF and PDF of a DED. An important observation is that the completion times of map tasks exhibit a delayed exponential function [8,9,10]. We will show that DED is nearly coincident with empirical data derived from prior research papers.

As illustrated in Figure 2.b, most mappers finish their tasks right after threshold, but a fraction of mappers finish their tasks only after a longer time. Since reducers can start only after completion of all their map tasks, they will be delayed because of these delayed mappers. Note that the intrinsic properties of architecture-level heterogeneity (e.g., big core versus small core) can further magnify the impact of stragglers. In this paper, we analytically model stragglers and show their effect on the end-to-end delay. We also optimize the delay using some schedulers depending on task mapping and number of mappers.



(a) **(b)**

Figure 2. CDF (a) and PDF (b) of DED.

3. Map Phase Modeling

This section gives the assumptions we make and explains the parameters used in our model. The DED is justified in each sub-section based on general knowledge and real observations, and the response time distribution of mapper nodes is proved analytically and verified via the available experimental data.

3.1 Assumptions

A MapReduce job (e.g., word count) can be seen as a set of tasks that must be completed to obtain the desire results (e.g., the frequency of each word). In our model, we assume that the different jobs are submitted to the system by some mean arrival rate (λ), or alternately, the submitted jobs to the system have a total mean inter-arrival time ($1/\lambda$). The mean arrival rate of the jobs can be interpreted as the average number of CPU instructions coming to the system per second, and is denoted by parameter λ as shown in Figure 1. A job comes to the MapReduce framework as a set of map tasks. In fact, the mapper nodes receive their map tasks with some probabilities p_1, p_2, ... p_{N_m} calculated by the scheduler. If N_m mapper nodes are involved, the job is split into $N_m M$ map tasks (where M is big enough). Alternately, given a scheduler, $p_i N_m M$ map tasks are sent to the i^{th} mapper node. In other words, the mean arrival rate of the tasks to the i^{th} mapper node is $\lambda_i = p_i\lambda$ where $\lambda = \sum_{i=1}^{N_m} \lambda_i$, and this arrival rate can capture the arrival rate of different jobs and the data skew.

Throughout the paper, the stochastic behavior of the i^{th} mapper node with its communication links is modeled as a *single queue* with a mean service rate of μ_i. The completion time of each mapper node is independent of the other mapper nodes, but the completion time of a map task may be dependent on the completion time of the other map tasks that reside in the same mapper node. Furthermore, the deterministic delay of mapper node computation and its communication links are captured by T_i, as defined in (3,4). Small stochastic variances across communication link delays add a nearly-deterministic delay overhead on all end-to-end path, because orchestrated traffic from mappers to reducers is nearly-deterministic, and the shuffling phase overlaps with the map phase of MapReduce. Some approaches [16,23] balance task mapping from mappers to reducers via pipelining or multi-level hash-based mechanism. Thus, the difference across the deterministic delays of the paths from mappers to a reducer is negligible without loss of generality. Table 1 shows the notation used in this paper.

Table 1. Notation used in our formulations.

Parameter	Notation	Explanation
Number of mapper nodes	N_m	The number of nodes in the datacenter assigned for map tasks
Mean inter-arrival time of i^{th} mapper's tasks	$1/\lambda_i$	The average time between the arrivals of tasks coming to i^{th} mapper
Mean service time of i^{th} mapper	$1/\mu_i$	The average servicing and routing time of the tasks by i^{th} mapper
Mean job inter-arrival time	$1/\lambda$	$\lambda = \sum_{i=1}^{N_m} \lambda_i$
Unit step function	$U(t)$	$U(t) = \begin{cases} 1; t \geq 0 \\ 0; t < 0 \end{cases}$
Dirac delta function	$\delta(t)$	$\delta(t) = dU(t)/dt$
Offset of DED for i^{th} mapper	T_i	The minimum amount of time required to complete a task
Response time of all map tasks	R_M	The required time to finish all map tasks by all mapper nodes
Response time of i^{th} mapper node	R_{M_i}	The required time to finish a task by i^{th} mapper node
Mean response rate of i^{th} mapper node	γ_i	The average required time to finish a task by i^{th} mapper node

3.2 Mapper Node as a Single Queue

A mapper node can be modeled as a FCFS (first-come first-served) infinite-buffer single queue. The distribution of response times of the map tasks is a function of the inter-arrival times of tasks as well as the service times of the mappers. In our model we investigate delayed-exponential service time. Later in Section 5, we study different distributions for task inter-arrival time and mapper node service time. Note that we are not restricted to a simple M/M/1 queue; we investigate the other inter-arrival times and service times that correspond to other potential scenarios in modern datacenters [27] such as Gamma or Erlang-k distributions as a general rational form of job inter-arrival time distribution in datacenters.

The clock rate of a mapper has a periodic characteristic and is proportional to the service rate μ_i as a linear coefficient (say C), i.e., sequential instructions can be executed successively with a time difference not less than $t_i \propto 1/(\mu_i C)$, and other instructions that involve memory or I/O requests can have an even longer time difference from the previous instruction. Consequently, the distribution of clock rate is $\delta(t - t_i)$, the Dirac delta function as $\delta(t) = dU(t)/dt$. The distribution of the service time for the instructions is exponential, as assumed by many prior papers [28,29,30]. The distribution of the combination of these two random variables is equivalent to the convolution ($*$) of these two distributions. The resulting service time distribution of a mapper node is thus a DED where its PDF can be expressed as:

$$\mu_i e^{-\mu_i t} U(t) *_{conv} \delta(t - t_i) = \mu_i e^{-\mu_i (t-t_i)} U(t - t_i), \quad (3)$$

where t_i is the minimum time required to process a CPU instruction. Naturally each map task contains a number of sequential instructions (say I), each having a minimum time to execute and route to a reducer. Therefore, the total service time of a map task is longer than $T_i = \sum_{j=1}^{I} t_j$ giving the offset of a DED in the service times. Note that the different mappers can have different service rates (μ_i) and offset times T_i which can make a datacenter heterogeneous, and in designing a scheduler for a heterogeneous datacenter, we only need these parameters (μ_i and T_i). As a result, the DED service time is a generalization of the exponential service time and we will show that it matches with real data.

3.3 Completion Time of Map Tasks

Response time (completion time) of map tasks is the waiting time in the buffer (queue) plus the time required to service them. In this section, we show that a sufficient condition for having a DED completion time of all map tasks to a reducer is to have a DED service time for each mapper using Lemma 1 and Lemma 2.

Lemma 1: A DED for service time results in nearly a DED for completion time (see Appendix A for the proof).

Lemma 2: If the distribution of completion time of map tasks in a mapper is a DED, the completion time of all map tasks to a reducer also has a nearly DED.

Proof: Assume that R_M is the total response time of all mapper nodes, i.e., the completion time of all map tasks, and R_{M_i} is the completion time of the i^{th} mapper node (where $1 \leq \mathrm{i} \leq N_M$) which has a DED CDF of the form $\left(1 - e^{-\gamma_i (t-T_i)}\right)\mathrm{U}(\mathrm{t} - T_i)$ or PDF as (3). Now, we have:

$$\begin{aligned}
&CDF(R_M) = P(R_M \leq t) \\
&= P\left(R_{M_1} \leq t \text{ or } R_{M_2} \leq t \ \ldots R_{M_{N_m}} \leq t\right) \\
&= 1 - P\left(R_{M_1} > t,\ R_{M_2} > t,\ \ldots,\ R_{M_{N_m}} > t\right) \\
&= 1 - P\left(R_{M_1} > t\right) P\left(R_{M_2} > t\right) \ldots P\left(R_{N_m} > t\right) \\
&= 1 - \prod_{i=1}^{N_m} P\left(R_{M_i} > t\right) \\
&= 1 - \prod_{i=1}^{N_m} \left(U(T_i - t) + e^{-\gamma_i (t-T_i)} U(t - T_i)\right) \\
&\approx \left(1 - e^{-\sum_{k=1}^{l} \gamma_{j_k} \left(t - T_{j_k}\right)}\right) U\left(t - T_{j_1}\right),
\end{aligned}$$

where $\{j_1, \ldots, j_{N_m}\}$ takes a subset of $\{1, \ldots, \mathrm{N_m}\}$ with respect to the position of t supposed to be $T_{j_1} \leq \cdots \leq T_{j_l} \leq \mathrm{t} \leq T_{j_{l+1}} \leq \cdots \leq T_{j_{N_m}}$. When T_i (offset) values are approximately equal, the final distribution will be a DED; otherwise, $CDF(R_M)$ can be expressed as piece-wise DED in periods between the T_i values. □

Table 2. Curve-fitting of the completion times of the published data to delayed exponential function.

Benchmark	Ref.	Exponential Rate (γ)	Least Square Error
Bing Search Engine	[8]	0.8945	0.0486
Facebook	[10]	0.0119	0.0418
Cloudera Customer (a)	[10]	0.0145	0.0557
Cloudera Customer (b)	[10]	0.0374	0.0200
Cloudera Customer (c)	[10]	0.0099	0.0316
Cloudera Customer (d)	[10]	0.0170	0.0364
Cloudera Customer (e)	[10]	0.0254	0.0450
OpenCloud	[9]	0.4591	0.0457
M45	[9]	0.2323	0.0566
WebMining	[9]	1.1543	0.0142

3.4 Validation of DED Completion Time

We show that our defined DED completion time for all map tasks coming to a reducer matches with the empirical completion times of map tasks that have been published in prior studies. We use the CDFs of published completion times of different MapReduce applications (Table 2) and try to fit a CDF of a DED on this data represented by the following function:

$$\left(1 - e^{-\gamma_i (t-T_i)}\right) U(t - T_i). \quad (4)$$

Note that the offset time T_i (constant for an application) can be directly taken from the empirical data, and it is only the mean response rate (γ_i) that we need to estimate. Newton's method for fast convergence has been used to fit a DED curve to the empirical curve that minimizes mean square error. As shown in Table 2, the least square error is not greater than 5% across all those workloads, strengthening our rationale for modeling completion times as a DED. Table 3 lists important parameters and their values used in our subsequent analysis and simulations. These values are based on data from the prior studies listed in Table 2.

Table 3. Parameter values used in experiments.

Parameter	Range
Mean inter-arrival time ($1/\lambda_i$)	0.1s-2s
Mean service time ($1/\mu_i$)	0.5s-2s
Offset time (T_i)	0.1s-100s
Mean response time ($1/\gamma_i$)	0.5s-1000s
Utilization ($\rho = \lambda_i/\mu_i$)	0.1-0.95

3.5 Mean Sojourn Time at a Reducer

Mean sojourn time (MST) at a reducer represents the average time required to synchronize all completed map tasks before a reducer can start its execution. MST can be a reasonable metric to represent the mean delay from job split to merge in a reducer as a fork-join queue for the first part of the end-to-end delay of a MapReduce job.

Definition 2 (Mean Sojourn Time): Given general CDFs of independent and identically distributed (i.i.d) response times of mapper nodes as $F_{R_{M_i}}(t) = P\left(R_{M_i} \leq t\right); i = 1 \ldots N_m$, by using *maximum order statistics (MOS)*, the required time for synchronization of the completed map tasks at i^{th} reducer (S_{R_i}) is the maximum of the response times of all mapper nodes, i.e., we can find $S_{R_i} = max(R_{M_1}, R_{M_2}, \ldots, R_{M_{N_m}})$ as:

$$F_{S_{R_i}}(t) = P\left(S_{R_i} \leq t\right) = \prod_{i=1}^{N_m} P(R_{M_i} \leq t) = \prod_{i=1}^{N_m} F_{R_{M_i}}(t). \quad (5)$$

The corresponding PDF can be easily expressed as follows:

$$f_{S_{R_i}}(t) = \frac{\partial}{\partial t} F_{S_{R_i}}(t) = F_{S_{R_i}}(t) \sum_{j=1}^{N_m} \frac{f_{R_{M_j}}(t)}{F_{R_{M_j}}(t)}. \quad (6)$$

Equations (7)-(9) given below are used in our future formulation. The expected value of sojourn time by using the inclusion–exclusion principle [31] can be expressed with respect to expected value of the minimum of random response times of every subset of mappers as:

$$\begin{aligned}
&MST = E\{S_{R_i}\} = E\left\{max\left(R_{M_1}, R_{M_2}, \ldots, R_{M_{N_m}}\right)\right\} \\
&= \sum_{i=1}^{N_m} \left\{ (-1)^{i+1} \sum_{\forall \{j_1, j_2, \ldots, j_i\} \subset \{1,2,\ldots,N_m\}} E\left\{min\left(R_{M_{j_1}}, R_{M_{j_2}}, \ldots, R_{M_{j_i}}\right)\right\} \right\}. \quad (7)
\end{aligned}$$

Furthermore, MST can be expressed in another way, with respect to the distribution of response time of each mapper node using (5) as:

$$MST = \int_{t=0}^{\infty} t f_{S_{R_i}}(t) dt = \int_{0}^{\infty} t \frac{\partial}{\partial t} \left(\prod_{i=1}^{N_m} P\left(R_{M_i} \leq t\right)\right) dt. \quad (8)$$

And, using expression (6), we have:

$$MST = \int_0^{\infty} t\, F_{S_{R_i}}(t) \left(\sum_{j=1}^{N_m} \frac{f_{R_{M_j}}(t)}{F_{R_{M_j}}(t)} \right) dt = \sum_{j=1}^{N_m} \int_0^{\infty} t\, F_{S_{R_i}}(t) \frac{f_{R_{M_j}}(t)}{F_{R_{M_j}}(t)} dt. \quad (9)$$

Note that the mean sojourn time (MST) is dependent on response time CDF of each mapper node ($P(R_{M_i} \le t)$) expressed by the inter-arrival time and service time of the mapper node. For heterogeneous queues (queues with different service times e.g. because of mappers with different performance) with a general form distribution of response time, we are not aware of any published closed-form formulations, bound or approximation for the MST. However, in the case of two homogeneous queues, there is an approximation and lower/upper bounds [32]. To our knowledge, the exponentially distributed response time of heterogeneous queues has no closed formula, no approximation, and no bound. In fact, only for the exponentially distributed response time of homogeneous queues (M/M/1 queues with the same μ), there are some approximations and boundaries in the statistics literature [33]. We derive the MST closed formula for the mapper nodes as M/M/1 queues using MOS in Lemma 3.

Lemma 3: Given M/M/1 mapper nodes with arrival rate λ_i and service rate μ_i where $i = 1 \ldots N_m$, using *maximum order statistics*, the MST of map tasks in a reducer is:

$$MST_{M/M/1} = \sum_{i=1}^{N_m} \left\{ (-1)^{i+1} \sum_{\forall \{j_1, j_2, \ldots, j_i\} \subset \{1,2,\ldots,N_m\}} \frac{1}{\sum_{k=1}^{i} (\mu_{j_k} - \lambda_{j_k})} \right\}. \quad (10)$$

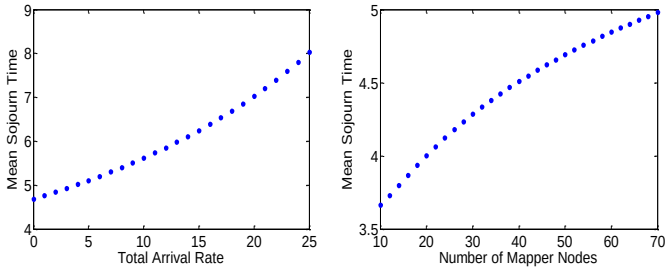
Proof: The response time of $j_k{}^{th}$ mapper node as an M/M/1 queue can be written as $P\left(R_{M_{j_k}} \le t\right) = \left(1 - e^{-(\mu_{j_k} - \lambda_{j_k})t}\right) U(t)$, and we have:

$$E\left\{min\left(R_{M_{j_1}}, R_{M_{j_2}}, \ldots, R_{M_{j_i}}\right)\right\} = \int_0^{\infty} tP\left(min\left(R_{M_{j_1}}, R_{M_{j_2}}, \ldots, R_{M_{j_i}}\right)\right) dt = \int_0^{\infty} t\left(1 - \prod_{k=1}^{i} P\left(R_{M_{j_k}} > t\right)\right) dt = \frac{1}{\sum_{k=1}^{i} (\mu_{j_k} - \lambda_{j_k})}.$$

Finally, by substituting this in (7), (10) will be obtained. □

Using validated DED response time in section 3, the sensitivity of the MST to the total mean arrival rate of the system can be derived and is shown in Figure 3.a, when the service rate of each mapper node is $\mu_i = 1$, and the number of mapper nodes is $N_m = 60$. When the arrival rate of the job increases, the mean sojourn time at a reducer homographically tends to infinity, if the number of mapper nodes is fixed. Figure 3.b gives an intuition about the variation of MST versus the number of mappers (N_m), when the service rate of each mapper node is $\mu_i = 1$, and the total arrival rate to the system is $\lambda = 2$. When the number of mapper nodes increases, if the total mean arrival rate is fixed, MST asymptotically tends to $O(log_e(N))$, where N is the number of mappers. Intuitively, there is a logarithmic algorithm to sync and merge completed tasks in one place, when a sync/merge operation can only happen between two completed tasks.

Based on the discussion above, we can conclude that *MST is the average time necessary to synchronize completed map tasks before the reduce phase*. As such, it can be expressed using the response times of each mapper. Our goal is to minimize MST with respect to task mapping and number of mappers.



(a) (b)

Figure 3. MST with respect to (a) total mean arrival rate and (b) the number of mapper nodes.

4. Potential Uses of the Model

Using an essential performance model, performance metrics of a parallel-running job such as response time or throughput can be optimized under power or budget constraints and a corresponding scheduler can be obtained. A good model should capture all important causal phenomena affecting the performance of the system as some key parameters and expressing the right effects of them in an easy manner. Using our model, a homogeneous or heterogeneous datacenter can be express using uniform ($\mu_1 = \mu_2 = \cdots = \mu_{N_m}$) versus non-uniform ($\mu_1 \neq \mu_2 \neq \cdots \neq \mu_{N_m}$) service rates (Figure 1). We envision three potential uses of our model with respect to architecture, system, and application parameters of a datacenter:

- One can model the performance parameters of nodes in a datacenter using inter-arrival times and service times of task trackers' queues more accurately, because it captures the deterministic behavior of task completion and the stochastic notion of the straggler problem.
- One can model the performance parameters of a MapReduce job as a fork-join network in a fashion that is more detailed than an M/M/1 queuing model but easier than a general distribution-based model. It can be used globally to improve the performance of a MapReduce job in a datacenter.
- Our model can also be used to optimize other target metrics such as throughput or utilization with power or budget constraints in a heterogeneous datacenter.

The delayed-exponential model of completion time of mappers is an extended version of the exponential model. It has the flexibility of being able to represent the arrival and service rates of different classes of workloads and cluster nodes. For example, the offset time parameter of a CPU-bound job is higher than that of a memory-bound job. Since the memory access times are random and take much longer than CPU access times, they can also be modeled by an exponential distribution. However, both jobs have a fixed delay of execution that cannot be reduced stochastically.

In addition, there is high degree of freedom to adjust departure rates of mappers to reducers ($\lambda_{i,j}$; $i = 1..N_m$ and $j = 1..N_r$) as shown in Figure 1, and we can solve linear independent equations to find reducers' arrival rates. By applying a DED completion time model to reducers, we can have similar MST formulations for reducers.

The exponential distribution model, having its highest value at zero, cannot capture the probability of two consecutive completed jobs with minimum inter-arrival time, because minimum inter-arrival time must be a positive value not zero. The DED response time model is also a good approximation of the response time of a typical datacenter server as a single queue, and can be employed in the analysis of different types of distributed computing networks. Based on Lemma 1, the DED completion time can be obtained using the DED service time, and this makes most of the analytical formulas much simpler.

Table 4. Summary of the lemmas.

Lemma	Explanation
1	DED service time → DED response time for a mapper node.
2	DED service time → DED completion time for all map tasks.
3	MST formula for M/M/1 mapper node
4	Joint optimization of MST is equivalent to separate optimizations.
5	Equilibrium property for D/D/1.
6	Equilibrium property for M/M/1.
7	Equilibrium property for G/M/1.
8	Sufficient conditions for optimal scheduling.
9	Optimal mapping from moments of the distribution.
10	Lower bound and upper bound of MST.
11	Optimal number of M/M/1 mapper nodes.
12	Optimal number of homogeneous mapper nodes.
13	Optimal number of mapper nodes for a fixed budget.

Since the focus of our paper is on performance, we use MST as an *end-to-end delay-aware metric*; other analyses may use different metrics depending on their focus. The potential metrics of interest could be power, budget, power/performance, or other combinations. First, the specification of the clusters and workloads should be evaluated. Considering Figure 2 and Expression (3), the service rate of the server and offset time related to the application can be obtained from the specifications. One can therefore derive a DED model for the service time. Then, the response time model can be obtained using the distribution of workload inter-arrival time that is discussed rigorously in the proof of Lemma 1 in Appendix A. We are interested in investigating the behavior of different schedulers via the single queue model of mappers with the DED response time when jobs come to the datacenter. This can be extended to a generalized multiple-queue model when we have multiple classes of jobs submitted to a datacenter. Table 4 gives a quick summary of the lemmas used in this work.

5. Map Phase Optimization

The delay in the critical path of a MapReduce job includes the delays in storage, network communications, map-task computation, synchronization of the map tasks, reduce-task computation, and aggregation of the reduce tasks. Heterogeneity in cluster resources makes the synchronization delay of map tasks higher. In fact, the problem of uneven map task completion times has been shown to be a serious impediment to the scalability of MapReduce computations [8,11,19]. While this has been studied experimentally, it has not been investigated from an analytical perspective. We want to minimize this synchronization delay by dividing the total arrival rate between mapper nodes. In this context, the unknown parameters are the residual arrival rate to each mapper node and the number of mapper nodes. The cost function in our problem is the mean sojourn time (MST) given by the task arrival rates of mapper nodes, and the sum of these arrival rates is a constant (mean job arrival rate), which is the constraint of the problem. The MST of the map tasks is the average time taken between the start of a map task and the time it reaches to a reducer. MST is the first moment (mean) of the distribution of completion times at a reducer. We are interested in reducing MST with respect to the number of mapper nodes and their task arrival rates.

Optimizing task arrival rates (say mapping) and the number of mapper nodes jointly is in general a hard problem. The JobTracker of MapReduce knows which mapper nodes are idle, i.e., it knows the state of the cluster. The task arrival rates of mapper nodes can be also controlled by the JobTracker to assign each mapper node the desired rate in mapping stage. The following lemma shows that the optimal mapping and the optimal number of mapper nodes can be *separately optimized*. Consequently, we can separate these two problems and solve each of them independently.

Lemma 4: Joint optimization of MST with respect to the number of mapper nodes and their task arrival rates is approximately equivalent to two separate optimizations, subject to, respectively, the number of mapper nodes and their task arrival rates. We have (see the proof in Appendix B):

$$\min_{N_m,\ \underline{\lambda}}(MST) = \min_{N_m}\left(\min_{\underline{\lambda}}(MST)\right) = \min_{\underline{\lambda}}\left(\min_{N_m}(MST)\right)$$
$$S.t.\ \lambda = \sum_{i=1}^{N_m}\lambda_i\,;\ \underline{\lambda} = \left[\lambda_1\lambda_2\lambda_3\ \ldots\lambda_{N_m}\right]^T. \quad (11)$$

5.1 Optimal Mapping

In this section, we investigate the optimal mapping of tasks to mapper nodes with respect to the mean sojourn time. Our analysis is carried out for different mapper node queues. We then use this to investigate efficient schedulers that can be employed to address the straggler problem.

5.1.1 Basic Schedulers

Hadoop fair job scheduler [5] divides and sends the same amount of work to each mapper node with task arrival rate λ_i and service rate μ_i for N_m mapper nodes, i.e., we have:

$$\lambda_1 = \lambda_2 = \lambda_3 = \cdots = \lambda_{N_m} = \lambda/N_m. \quad (12)$$

The fair job scheduler is optimal when we have a completely homogeneous cluster. The "*no bottleneck system necessary condition"* means that the mean job arrival rate should be less than the total service rate of the mapper nodes ($\lambda < \sum_{i=1}^{N_m}\mu_i$). Depending on the scheduling algorithm employed, we may have some tighter bounds, as in this case no bottleneck node necessary condition is:

$$\lambda < N_m\mu_{min}. \quad (13)$$

A fair job scheduler (like the Hadoop scheduler) makes the mean task arrival rates equal for all mapper nodes as captured by Expression (12), i.e., it gives each mapper node the same amount of work. Figure 4 plots the result for a cluster with the same number (from 40 to 200) of low-performance (service time $s_i = 1.25$) and high-performance ($s_i = 1$) nodes. The MST of the fair job scheduler in Expression (12), with respect to the total arrival rate and the number of M/M/1 heterogeneous mapper nodes, always increases when the total arrival rate increases. However, the MST with respect to the number of nodes has a minimum, i.e., there is an optimal number of nodes given a job arrival rate. If the job arrival rate increases, this minimum number of nodes also increases.

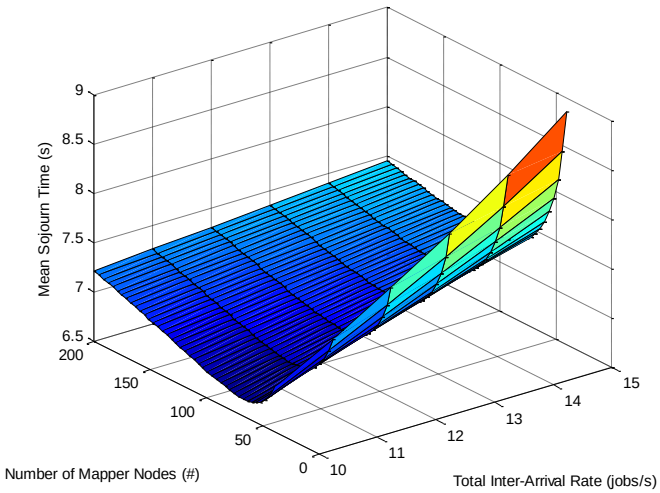


Figure 4. MST with respect to the total arrival rate and the number of M/M/1 mapper nodes in a fair job scheduler.

We have implemented a queueing-based framework using CSIM (an efficient multi-thread queueing system simulator [34]) and rational parameters in the range of real parameters (Table 3). Our queueing-based model takes the distribution of the inter-arrival time and the DED of service time as inputs and gives the mean service time, mean queue length of all queues, utilization, throughput, and mean response time as outputs.

In the first simulation, the same number of low-performance (service time $s_i = 1.5$) and high-performance ($s_i = 1$) mapper nodes exist. The simulation results plotted in Figure 5 are for these heterogeneous mappers with the fair job scheduler. They match with analytical results in Figure 4 and indicate that the throughput of the nodes in the cluster remains almost the same, but the other parameters such as utilization, queue length and response time, change based on the inter-arrival time to the system and service time of mapper nodes.

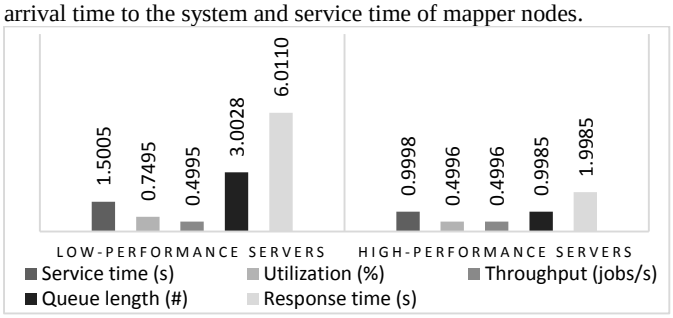


Figure 5. The mean value of important parameters of a heterogeneous datacenter with the fair job scheduler.

Next, we define a metric that can be used to mathematically compare the performance of the different types of schedulers. We start by giving the definition of the optimal MST.

Definition 3 (Optimal mapping based on MST): Optimal mapping for a known number of mapper nodes (N_m) finds the optimal values of the mean task arrival rates ($\lambda_1, \lambda_2, \dots, \lambda_{N_m}$) making MST minimum. The constraint here is the mean job arrival rate to the system (λ). The service rate of each mapper node is assumed to have a known distribution with a mean of μ_i. In mathematical terms, we have:

$$\min_{\underline{\lambda}}(MST) = \min_{\underline{\lambda}} \int_0^{\infty} t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m} F_{R_{M_i}}(t)\right) dt \qquad S.t.\ \lambda = \sum_{i=1}^{N_m} \lambda_i\ ;\ \underline{\lambda} = \left[\lambda_1 \lambda_2 \lambda_3\ \dots \lambda_{N_m}\right]^T, \tag{14}$$

where $F_{R_{M_i}}(t)$ is the CDF of the response time of the i^{th} mapper node as a function of μ_i and λ_i.

The optimal solution of the mapping problem is derived by using the Lagrange Multipliers method and solving the following set of non-linear equations:

$$\nabla_{\underline{\lambda},\alpha}\left(MST - \alpha\left(\lambda - \sum_{i=1}^{N_m} \lambda_i\right)\right) = 0, \tag{15}$$

where $\nabla_{\underline{\lambda},\alpha}(.)$ is the multi-dimensional gradient operator with respect to $\underline{\lambda}$ and α. We now give the following definition as a property for having optimal MST.

Definition 4 (Equilibrium Property): To optimize the mapping of the tasks, the set of non-linear equations derived from (15) have the Equilibrium Property (EP) with the linear constraint $\lambda = \sum_{i=1}^{N_m} \lambda_i$. Solving the set of non-linear equations gives the optimal solution for $\underline{\lambda}$. The equilibrium property can be expressed as:

$$\frac{\partial MST}{\partial \lambda_1} = \frac{\partial MST}{\partial \lambda_2} = \cdots = \frac{\partial MST}{\partial \lambda_{N_m}}. \tag{16}$$

The above property cannot be easily solved or practically used to optimize MapReduce job scheduling. It is an N_m-dimensional non-linear optimization problem with a linear constraint. We will shortly present several sufficient conditions to satisfy equilibrium property in Lemma 8. The following lemmas handle certain special cases where Equation (9) can be easily solved.

Lemma 5: Given Def.2-4 for D/D/1 mapper nodes ($i = 1 \dots N_m$), equilibrium property can be expressed as follows:

$$\frac{\mu_1}{\lambda_1} = \frac{\mu_2}{\lambda_2} = \cdots = \frac{\mu_{N_m}}{\lambda_{N_m}}. \tag{17}$$

We call the above property the deterministic equilibrium property (D-EP), and the optimal solution for λ_is is:

$$\lambda_i = \lambda \frac{\mu_i}{\sum_{j=1}^{N_m} \mu_j}\ ; i = 1 \dots N_m. \tag{18}$$

The "no bottleneck node necessary" condition is:

$$\lambda < \frac{\mu_{min}}{\mu_{max}} \sum_{i=1}^{N_m} \mu_i. \tag{19}$$

Proof: Equation (17) can be derived by making the deterministic response times of all mappers equal using:

$$\frac{\mu_1}{\lambda_1} = \frac{\mu_2}{\lambda_2} = \cdots = \frac{\mu_{N_m}}{\lambda_{N_m}} = \frac{\sum_{j=1}^{N_m} \mu_j}{\lambda}.$$

Then, Equation (18) is deduced, and $\lambda < \mu_i\ ;\ \forall i$ will give (19). □

Fair queue, shortest queue first or μ-proportional scheduling make queue length equal and are the same as the pure-deterministic scheduling in Equation (17) as we have:

$$Q_1 = Q_2 = \cdots = Q_{N_m} \Leftrightarrow \rho_1 = \rho_2 = \cdots = \rho_{N_m} \tag{20}$$

Considering $T_i \propto \lambda_i/\mu_i$, i.e., the offset of the DED response time has a linear relationship with λ_i/μ_i, we can say that pure-deterministic scheduling and shortest queue first scheduling are equivalent. This is because we have:

$$\frac{\lambda_1}{\mu_1} = \frac{\lambda_2}{\mu_2} = \cdots = \frac{\lambda_{N_m}}{\mu_{N_m}} \Leftrightarrow T_1 = T_2 = \cdots = T_{N_m}. \tag{21}$$

Figure 6 plots the MST of the pure-deterministic scheduler with respect to the total arrival rate and the number of M/M/1 heterogeneous mapper nodes. The MST growth given by the total arrival rate is homographic versus the MST growth given by the number of mapper nodes is $O(ln(n))$. Comparing Figure 4 with Figure 6, one can see that the pure-deterministic scheduler has a lower MST with respect to the fair job scheduler.

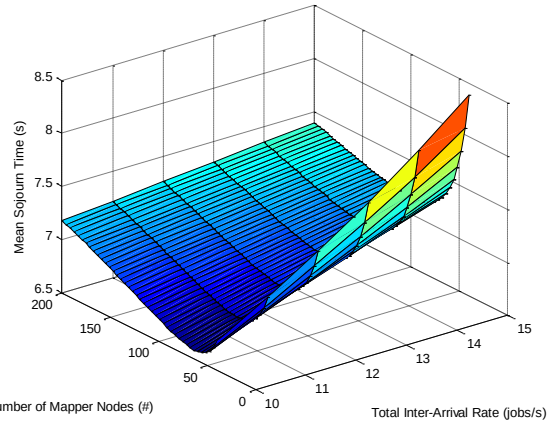


Figure 6. MST with respect to the arrival rate and the number of M/M/1 mapper nodes for the pure-deterministic scheduler.

Figure 7 shows the simulation results for heterogeneous servers with the pure-deterministic scheduler. One can observe from these results that pure-deterministic scheduler keeps the utilization and queue length of the different type nodes equal as in (20, 21) but response time and throughput are different, which makes the MST higher.

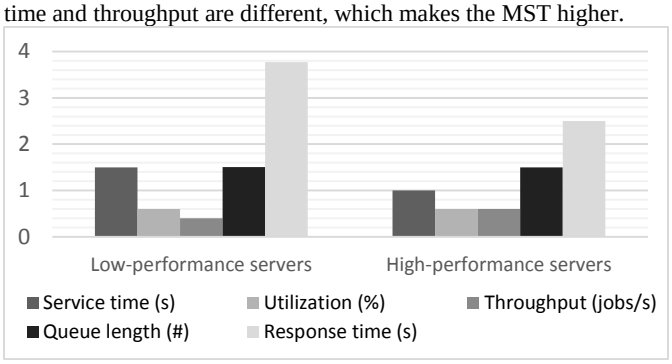


Figure 7. Mean values of important parameters of a heterogeneous datacenter with the pure-deterministic scheduler.

Lemma 6: Given Def.2-4 for M/M/1 mapper nodes (i = 1 ... N_m), the equilibrium property would be (see the proof in Appendix C):

$$\mu_1 - \lambda_1 = \mu_2 - \lambda_2 = \cdots = \mu_{N_m} - \lambda_{N_m}. \quad (22)$$

We call above property stochastic equilibrium property (S-EP), and the optimal λ_is are:

$$\lambda_i = \mu_i + \frac{\lambda - \sum_{j=1}^{N_m} \mu_j}{N_m} \; ; i = 1 \dots N_m. \quad (23)$$

Without having a bottleneck system as the inequality $\lambda < \sum_{i=1}^{N_m} \mu_i$, there is no bottleneck node necessary condition for any mapper node, but we may have negative values for λ_i, i.e., some jobs should be migrated from slow nodes (i) to other nodes. Thus, the no negative λ_i necessary condition is:

$$\sum_{i=1}^{N_m} (\mu_i - \mu_{min}) < \lambda. \quad (24)$$

Q-proportional scheduling and pure-stochastic scheduling are similar, because:

$$\frac{\lambda_1}{Q_1} = \frac{\lambda_2}{Q_2} = \cdots = \frac{\lambda_{N_m}}{Q_{N_m}} \Leftrightarrow 1/(\mu_1 - \lambda_1) = \cdots = 1/(\mu_{N_m} - \lambda_{N_m}). \quad (25)$$

Figure 8 plots the MST of the pure-stochastic scheduler with respect to the total arrival rate and the number of M/M/1 (heterogeneous) mapper nodes. The MST growth trend is similar to that of the other scheduler. Comparing Figure 8 against Figures 4 and 6, one can observe that the pure-stochastic scheduler has a lower MST, given any total arrival rate and any number of nodes.

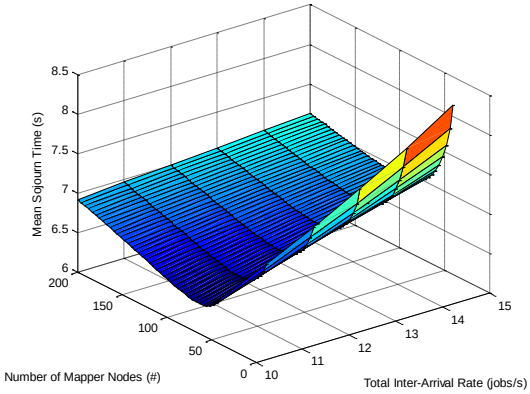


Figure 8. MST with respect to the arrival rate and the number of M/M/1 mapper nodes for the pure-stochastic scheduler.

Figure 9 plots the simulation results for heterogeneous mapper nodes with the pure-stochastic scheduler. In this case, the mean response time of all nodes are nearly equal and the minimum delay can be achieved for the system.

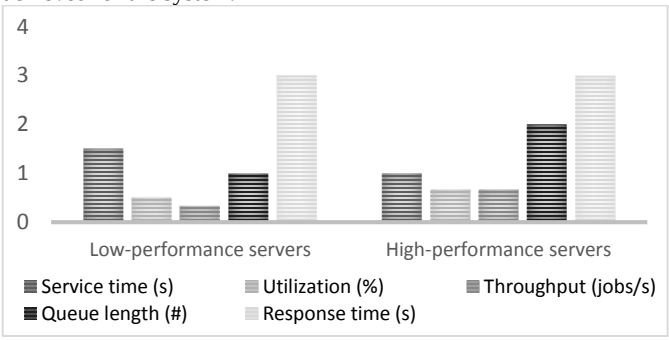


Figure 9. Mean values of important parameters of a heterogeneous datacenter with the pure-stochastic scheduler.

For the M/M/1 mapper nodes, pure-stochastic mapping has a lower MST compared to the fair job and pure-deterministic mappings shown in Figure 10. The MST of the pure-stochastic strategy is lower than the others even under very low arrival rates.

5.1.2 Advanced Schedulers

In this section, we investigate more sophisticated schedulers based on some general job arrival rates and service rates. Some of the results are stochastically achievable, and we give two algorithms that converge to the optimal task arrival rate for each mapper node.

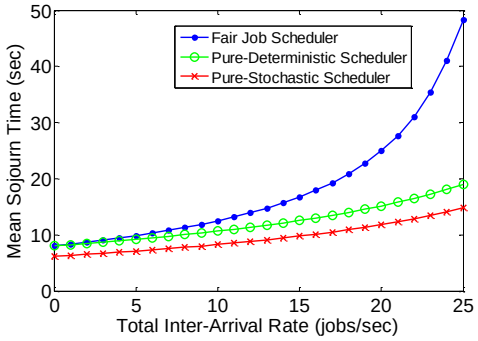


Figure 10. MST comparison between fair job, pure-deterministic, and pure-stochastic schedulers.

Lemma 7: Given Def.2-4 for G/M/1 mapper nodes (i = 1 ... N_m) the equilibrium property would be as follows (proof in Appendix D):

$$\mu_1(1 - r_{0_1}) = \mu_2(1 - r_{0_2}) = \cdots = \mu_{N_m}\left(1 - r_{0_{N_m}}\right), \quad (26)$$

where r_{0_i} is the unique real root of $z = A_i^*[\mu_i(1-z)]$ in (0,1) and A_i^* is the LST (Laplace–Stieltjes transform) of $A_i(t)$ (CDF of inter-arrival time). The solution of MST for the G/M/1 mapper nodes is not trivial and the following algorithm shows how we can find the optimal inter-arrival times.

Algorithm 1: Given G/M/1 mapper nodes, Alg. 1 finds CDFs of optimal inter-arrival time (A_i) for the equilibrium property in (26).

Given Def.2, the LST of A_i is the function of the mean task inter-arrival time (λ_i). Let $A_i^* = g_i(\lambda_i)$. Also $g_i^{-1}(A_i^*)$ exists in $(0, \lambda)$ where λ is total mean arrival rate of the system, e.g. in M/M/1 case we have $A_i^*(s) = \lambda_i/(\lambda_i + s)$ and $\lambda_i = A_i^* s/(1 - A_i^*)$. The cost function that we are interested to find the root of by *Secant Method* [35] with small error ϵ is as follows:

$$f(\alpha) = \lambda - \sum_{i=1}^{N_m} g_i^{-1}(A_i^*(\alpha)) \quad (27)$$

1. Let $\alpha_0 = 0.5\mu_{\min},\ \alpha_1 = \mu_1\big(1 - g_1(\lambda/N_m)\big), n = 1$
2. $n = n + 1$
3. $\alpha_n = \alpha_{n-1} - \frac{f(\alpha_{n-1})(\alpha_{n-1}-\alpha_{n-2})}{f(\alpha_{n-1})-f(\alpha_{n-2})}$
4. if $(|f(\alpha_n)| > \epsilon)$ then go to 2, else $\alpha = \alpha_n$.

Before deriving the corresponding expressions for other distributions, let us first give the sufficient conditions to achieve the optimal mapping.

Lemma 8: Given Def.2-4, if one of the following sufficient conditions is satisfied, the solution of the equations set of the sufficient condition and linear constraint ($\lambda = \sum_{i=1}^{N_m} \lambda_i$) is guaranteed to be optimal.

$$\forall t \in \mathcal{R}^+, \forall i,j \in \{1,2,\dots,N_m\}$$
$$\frac{1}{F_{R_{M_i}}}\frac{\partial}{\partial\lambda_i}F_{R_{M_i}}(t) = \frac{1}{F_{R_{M_j}}}\frac{\partial}{\partial\lambda_j}F_{R_{M_j}}(t) \quad (28)$$
$$F_{R_{M_i}}(t) = F_{R_{M_j}}(t) \quad (29)$$
$$M_{R_{M_i}}(t) = M_{R_{M_j}}(t);\ \ M_X(t) = E\{e^{tX}\} \quad (30)$$
$$\forall s \in \mathcal{R}, \forall i,j \in \{1,2,\dots,N_m\}: W_i^*(s) = W_j^*(s), \quad (31)$$

where $F_{R_{M_i}}(t)$ is CDF, $M_{R_{M_i}}(t)$ is moment-generating function, and $W_i^*(s)$ is the LST of the response time of the i^{th} mapper node (see the proof in Appendix E).

Also, for M/G/1 mapper nodes with arrival rate λ_i and service time distribution $g_i(t)$ $(E\{g_i(t)\} = \mu_i)$, the sufficient condition to have optimal mapping by inspiring from (31) is:

$$\frac{\left(1-\frac{\lambda_i}{\mu_i}\right)sg_i(s)}{s-\lambda\big(1-g_i(s)\big)} = \frac{\left(1-\frac{\lambda_j}{\mu_j}\right)sg_j(s)}{s-\lambda\left(1-g_j(s)\right)}, \quad (32)$$

where $g_i(s)$ and $W_i^*(s)$ are respectively the LST of $g_i(t)$ and the response time of i^{th} mapper node. This conclusion can be easily derived from the Pollaczek-Khintchine transform. Also for G/G/1 mapper nodes with LST of inter-arrival time distribution $A_i^*(s)$ and LST of service time distribution $B_i^*(s)$, the sufficient condition to have optimal mapping by inspiring from equation (31) is:

$$W_{q_i}(0)B_i^*(s) + \left(1 - W_{q_i}(0)\right)B_i^*(s)W_{q_i}^*(s) = W_{q_j}(0)B_j^*(s) + \left(1 - W_{q_j}(0)\right)B_j^*(s)W_{q_j}^*(s), \quad (33)$$

whose the term $W_{q_i}^*(s)$ is the LST of waiting time distribution and satisfies Lindley's equation $W_{q_i}(t) = -\int_0^\infty W_{q_i}(x)dU_i(t-x)$ such that $U_i(x) = \int_{max(0,x)}^\infty B_i(t)dA_i(t-x)$.

Corollary 9: Given Def.2-4 of MST problem, if we have the moments of the distribution of task inter-arrival time $(A(t))$, the optimal mapping exists.

Proof: Similar to (26) and (27) you can assume the following relation between the LST of task inter-arrival times $(A_i^*(s))$ and the LST of response times $(W_i^*(s))$:

$$W_i^*(s) = f\big(A_i^*(s)\big);\ \forall s \in \mathcal{R}, \forall i \in \{1,2,\dots,N_m\}, \exists f \quad (34)$$

At this point, we just need to adjust the moments of inter-arrival times to satisfy Equation (31) and the constraint while we have:

$$1 + \frac{s}{1!}W_i^{*\prime(0)} + \frac{s^2}{2!}W_i^{*\prime\prime(0)} + \cdots = f\left(1 + \frac{s}{1!}A_i^{*\prime}(0) + \frac{s^2}{2!}A_i^{*\prime\prime}(0) + \cdots\right) \quad (35)$$

As the number of unknown variables $(A_i^*(s))$ and the related equation (35) and the linear constraint are equal, the optimal mapping exists.□

In the next lemma, we obtain results which are more suitable to be employed in practice. It is difficult and time-consuming to find the best mapping of inter-arrival time distributions to have such equilibrium properties. Instead of that, one can imagine making the first moment (mean) of the response time distribution equal. It is equivalent to make the first terms of Taylor's series of the LST of response time distribution $(W_i'(s=0))$ equal. The lemma below tries to show the boundaries of the MST problem and an approximate solution which is very useful for most of the cases.

Lemma 10: Given Def.2-4 the lower bound and upper bound of MST can be found as follows:

$$max\left(E\{R_{M_1}\}, E\{R_{M_2}\}, \dots, E\left\{R_{M_{N_m}}\right\}\right) \le MST \le \sum_{i=1}^{N_m} E\{R_{M_i}\} \quad (36)$$

And the equilibrium property of lower bound can be expressed as:

$$E\{R_{M_1}\} = E\{R_{M_2}\} = \cdots = E\left\{R_{M_{N_m}}\right\}. \quad (37)$$

We call this the means equilibrium property (M-EP) and approximate optimal scheduler can have the M-EP. But derivative of means equilibrium property (DM-EP) optimizes MST upper bound:

$$\frac{d}{d\lambda_1}E\{R_{M_1}\} = \frac{d}{d\lambda_2}E\{R_{M_2}\} = \cdots = \frac{d}{d\lambda_{N_m}}E\left\{R_{M_{N_m}}\right\}. \quad (38)$$

See the proof in Appendix F.

M-EP approximately optimizes MST and gives sub-optimal mean arrival rates ($\underline{\lambda}$) by making the first moments of response times equal. The only degrees of freedom are the mean arrival rates ($\underline{\lambda}$), so in an approximate approach, we can only have the M-EP and not the other higher moments equal. This approximation is correct when the other moments of the response time of mapper nodes are negligible or the deviation in (30) is insignificant, i.e., when there is no bottleneck node in the system. In fact the gap between the optimal solution and this approximate solution is insignificant. We now present the algorithms to have M-EP for DED/DED/1, M/G/1 and G/G/1 mapper nodes.

The mean response time of a DED mapper node can be expressed as $T_i + 1/(\mu_i - \lambda_i)$ where $T_i = D\lambda_i/\mu_i$ for some constant D. Then the M-EP for DED mapper nodes would be:

$$\forall i,j \in \{1,2,\dots,N_m\}: T_i + 1/(\mu_i - \lambda_i) = T_j + 1/(\mu_j - \lambda_j) \quad (39)$$

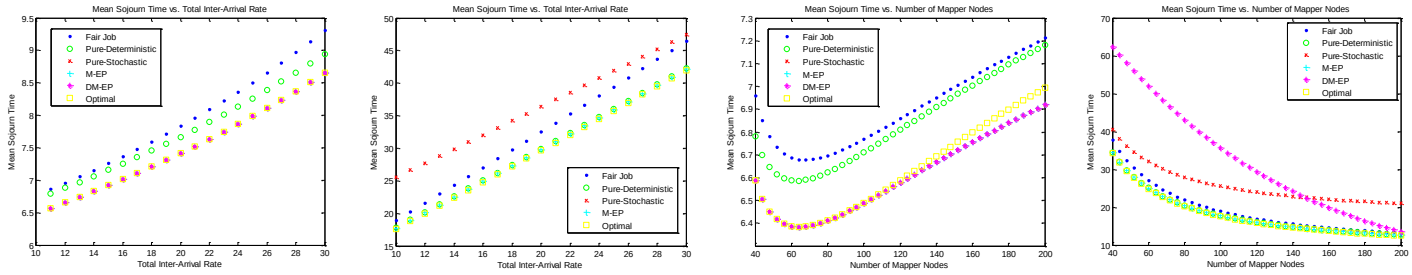
Algorithm 2: Given DED mapper nodes Alg. 2 will find sub-optimal arrival rates (λ_i) for the equilibrium property in (39), where it is just needed to iterate Alg.1 with the cost function as:

$$f(\alpha) = \lambda - \sum_{i=1}^{N_m} \frac{(\alpha + D)\mu_i - \sqrt{\big(\mu_i(\alpha - D)\big)^2 + 4D\mu_i}}{2D} \quad (40)$$

M-EP for M/G/1 mapper nodes is:

$$\forall i,j \in \{1,2,\dots,N_m\}$$
$$\frac{\lambda_i\left(\sigma_{B_i}^2 + \frac{1}{\mu_i^2}\right)}{2\left(1-\frac{\lambda_i}{\mu_i}\right)} + \frac{1}{\mu_i} = \frac{\lambda_j\left(\sigma_{B_j}^2 + \frac{1}{\mu_j^2}\right)}{2\left(1-\frac{\lambda_j}{\mu_j}\right)} + \frac{1}{\mu_j}, \quad (41)$$

where λ_i, μ_i, and $\sigma_{B_i}^2$ are respectively the i^{th} mapper node's arrival rate, mean service rate, and variance of service rate.



(a) (b) (c) (d)

Figure 11. MST comparison of different schedulers with respect to total arrival rate and number of mappers for (a) memory-bound job ($D \approx 0$), (b) CPU-bound job ($D = 100$), (c) memory-bound job ($D \approx 0$), and (d) CPU-bound job ($D = 100$).

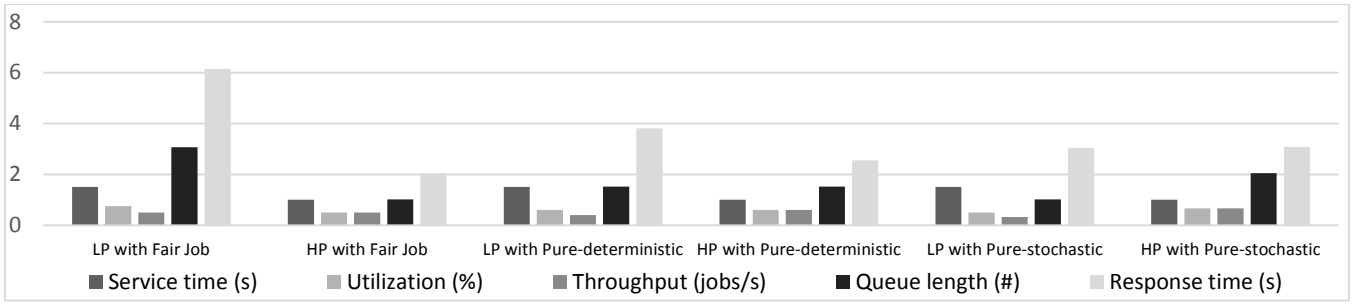


Figure 12. Comparison of different schedulers' parameters (LP = Low performance, HP = High performance)

Algorithm 3: Given M/G/1 mappers, Alg. 3 finds sub-optimal arrival rates (λ_i) for the equilibrium property in (41), where it is just needed to iterate over Alg.1 with the following cost function:

$$f(\alpha) = \lambda - \sum_{i=1}^{N_m} 1/\left(\frac{\sigma_{B_i}^2 + \frac{1}{\mu_i^2}}{2\left(\alpha - \frac{1}{\mu_i}\right)} + \frac{1}{\mu_i}\right) \tag{42}$$

Using Kingsman's approximation, M-EP for G/G/1 is:

$$\forall i, j \in \{1,2,\dots,N_m\}$$
$$\frac{\frac{\lambda_i}{\mu_i}\left(\frac{\sigma_{A_i}^2}{\lambda_i^2} + \frac{\sigma_{B_i}^2}{\mu_i^2}\right)}{2(\mu_i - \lambda_i)} + \frac{1}{\mu_i} = \frac{\frac{\lambda_j}{\mu_j}\left(\frac{\sigma_{A_j}^2}{\lambda_j^2} + \frac{\sigma_{B_j}^2}{\mu_j^2}\right)}{2(\mu_j - \lambda_j)} + \frac{1}{\mu_j} \tag{43}$$

Algorithm 4: Given G/G/1 mappers for the defined problem in Def. 2-4, Alg. 4 will find sub-optimal mean arrival rates (λ_i) for the equilibrium property in (43), where it is just needed to iterate Alg.1 with knowing $\forall i: \sigma_{A_i}^2/\lambda_i^2 = \sigma_A^2/\lambda^2$ by the cost function as follows:

$$f(\alpha) = \lambda - \sum_{i=1}^{N_m} \mu_i/\left(1 + \frac{\frac{1}{\mu_i}\left(\frac{\sigma_A^2}{\lambda^2} + \frac{\sigma_{B_i}^2}{\mu_i^2}\right)}{2\left(\alpha - \frac{1}{\mu_i}\right)}\right) \tag{44}$$

In the following graphs (Figures 11a-11d) we use our DED model of the map phase for a CPU-bound job (when T_i as the deterministic coefficient is big, i.e. $D \gg 1$) and a memory-bound job (when T_i is small, i.e. $D \approx 0$). The graphs show the comparison between different schedulers as mentioned before with respect to total mean arrival rate and number of heterogeneous mapper nodes.

When the target MapReduce job is memory-bound or has random memory access, it is more stochastic and the deterministic coefficient for this type of job is close to zero. For memory-bound jobs, the optimal scheduler that tries to find λ_is minimizing MST is nearly coincident with pure-stochastic, M-EP, and DM-EP schedulers. For CPU-bound jobs, the optimal scheduler is nearly coincident with pure-deterministic and M-EP schedulers. As a result, M-EP is a good approximation of optimal solution in this model. When the input job is CPU-bound, the pure-stochastic scheduler cannot track the optimal response time well. Some schedulers sometimes outperform the non-negative optimal scheduler, because they output negative λ_i. The non-negative optimal scheduler always gives positive not-migratory λ_i, but in a non-negative region the optimal scheduler outperforms the others.

Same scenario in simulation results (Figure 12) for $D \approx 0$ indicates that the pure-stochastic scheduler can make all nodes' response times nearly equal, and the other schedulers' deviation is higher. Since MST is a function of the slowest node, here M-EP is near to the pure-stochastic scheduler, and it verifies that our approach to equalize the response time of mappers getting a better MST.

5.2 Optimal Number of Mapper Nodes

It is interesting to note that there is an optimal number of mapper nodes that makes the mean sojourn time minimum. Intuitively, when we increase the number of mapper nodes in the system the amount of work shared among mappers decreases. As a result, the smaller jobs can be completed sooner. However, blindly increasing the number of mapper nodes is not a good strategy, because more mapper nodes need more time to be synced. Therefore, there is a *tradeoff* between smaller job completion time and higher sync time. A general closed form solution for the optimal number of mapper nodes is very difficult. We can use the result of Lemma 4 and assume that the optimal mapping makes some of the moments of the response time distribution equal. Thus, we can say that mapper nodes' distributions are approximately similar.

Definition 5 (Optimal Number of Mapper Node): Given Def.2-4 the optimal number of mapper nodes is an integer number that makes the mean sojourn time minimum. If $F(t)$ is CDF of response time of any mapper node, then we have:

$$\begin{aligned} \min_{N_m}(MST) &= \min_{N_m} \int_0^\infty t \frac{\partial}{\partial t}\left(F^{N_m}(t)\right)dt \\ &= \min_{N_m} N_m \int_0^\infty t f(t) F^{N_m - 1}(t) dt \end{aligned} \tag{45}$$

Lemma 11: Given Def.5 for M/M/1 mapper nodes with μ as the service rate and λ as the total arrival rate to the system, the optimal number of mapper nodes to minimize MST is equal to the minimum value of the following function with respect to n:

$$f(n) = \frac{1}{\mu - \frac{\lambda}{n}} \sum_{i=1}^{n} \left((-1)^{i+1} \frac{n!}{(n-i)!\,(i-1)!} \right) \quad (46)$$

Proof: In (10) if we make all $\mu_{j_k} - \lambda_{j_k} = \mu - \lambda/N_m$ equal, we have:

$$\begin{aligned} & MST_{M/M/1} \\ &= \sum_{i=1}^{N_m} \left\{ (-1)^{i+1} \sum_{\forall \{j_1, j_2, \ldots, j_i\} \subset \{1,2,\ldots,N_m\}} \frac{1}{\sum_{k=1}^{i} (\mu_{j_k} - \lambda_{j_k})} \right\} \\ &= \sum_{i=1}^{N_m} \left\{ (-1)^{i+1} \sum_{\forall \{j_1, j_2, \ldots, j_i\} \subset \{1,2,\ldots,N_m\}} \frac{1}{i\left(\mu - \frac{\lambda}{n}\right)} \right\} \\ &= \frac{1}{\mu - \frac{\lambda}{N_m}} \sum_{i=1}^{N_m} \left\{ (-1)^{i+1} \frac{1}{i} \binom{N_m}{i} \right\} \end{aligned}$$

When N_m is the parameter, it is equivalent with (46). □

Figure 13 shows the existence of a knee point for the optimal number of mapper nodes in a heterogeneous datacenter. In this result, we have different nodes with uniform service rates of 1, 0.9, 0.8, and 0.7. The number of nodes is varied between 40 and 200. Also, the arrival rate is 10 and offset time is set to 3.5.

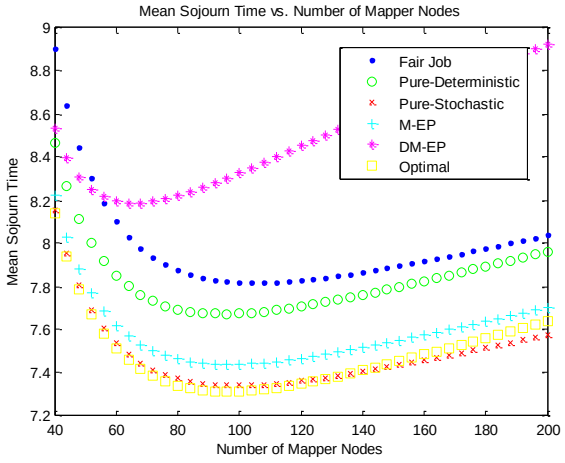


Figure 13. Comparison of optimal number of heterogeneous mapper nodes based on MST with $D = 3.5$.

Lemma 12: Given Def.5 for homogeneous mapper nodes with μ as the service rate and λ as the total arrival rate to the system, the optimal number of mapper nodes is equal to the minimum of the following function with respect to n:

$$f(n) = \frac{1}{\mu - \frac{\lambda}{n}} \left[H_n + \left(\left(\sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} \sum_{j=1}^{i} \binom{i}{j} \frac{(j-1)!}{i^{j+1}} \right) - H_n \right) \frac{\lambda}{n\mu} \right] \quad (47)$$

where $H_n = \sum_{i=1}^{n} 1/i$.

Proof: This is a direct result of the approximation of MST in [36]. It is based on the interpolation of lower bound and upper bound for low/medium/high traffic and can be approximately used for a general form distribution. We can find the optimal number of mapper nodes by refining it as Expression (47). □

Lemma 13: Given Def.5 for M/M/1 mapper nodes with K different service rates, prices, and numbers as (μ_1, P_1, N_1), (μ_2, P_2, N_2), … (μ_K, P_K, N_K). The total arrival rate to the system is λ and the budget constraint is $Budget = \sum_{i=1}^{K} P_i N_i$ where $N_m = \sum_{i=1}^{K} N_i$. The optimal number of mapper nodes of each type to minimize MST is equal to the minimum of the following function with respect to $N_1, N_2, \ldots, N_K$:

$$f(N_1, \ldots, N_K) = \frac{\sum_{i=1}^{K} N_i}{\sum_{i=1}^{K} (N_i \mu_i) - \lambda} \sum_{i=1}^{N_m} \left((-1)^{i+1} \binom{N_m}{i} / i \right) \quad (48)$$

The proof is straightforward like the proof of Lemma 11. One can derive a similar corollary for the general case with respect to Lemma 12. Figures 14-16 give the optimal number of Low-performance (LP) and high-performance (HP) servers in a heterogeneous datacenter with a fixed amount of budget for low/medium/high traffic. The service rate ratio and price ratio of LP to HP are based on [37]. The budget is changing between 2000 and 40000 where price/service rate for LP servers and HP servers are respectively 400/1 and 800/1.25, and job arrival rates for low/medium/high traffic are respectively 1/5/25.

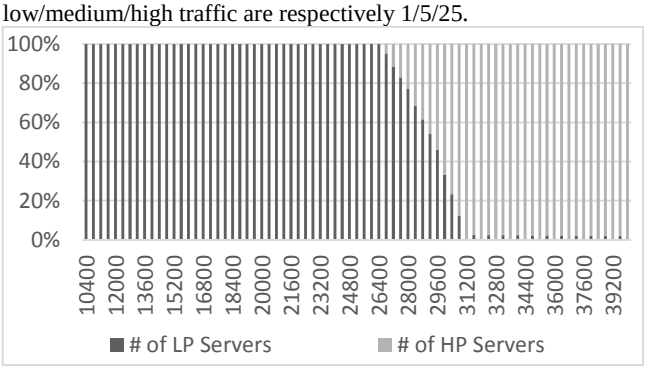


Figure 14. Stack ratio of LP servers to HP servers with respect to Budget in high traffic.

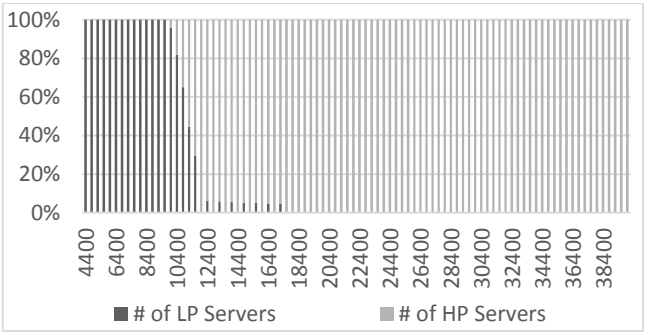


Figure 15. Stack ratio of LP servers to HP servers with respect to Budget in medium traffic.

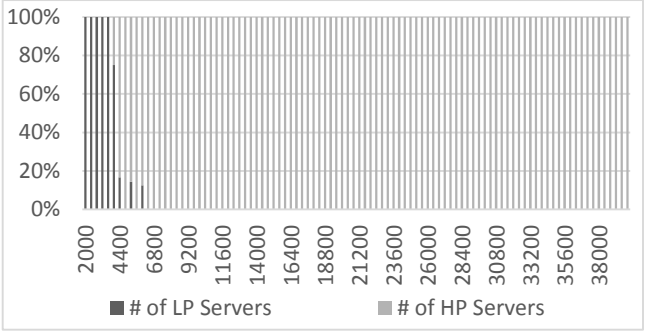


Figure 16. Stack ratio of LP servers to HP servers with respect to Budget in low traffic.

6. Concluding Remarks

Targeting MapReduce applications, in this paper, we model the service time of mapper nodes as a single queue with the delayed exponential distribution (DED), and also show that their response time has a similar behavior. Next, using this analytical result, we model the map phase of a single-pass MapReduce job and formulate the mean sojourn time (MST) at a reducer node by means of task arrival rates and service rates of mapper nodes. MST is a potential metric for optimizing end-to-end delay in a MapReduce framework. Based on different types of inter-arrivals and service rates, we optimize the MST parameter and investigate the equilibrium property for different types of queues. To realize the minimum map phase delay in a heterogeneous datacenter, we have also investigated different types of schedulers. Our results show that the optimal scheduler is better than any deterministic/stochastic scheduler, and it gives the optimal mapping of the job and number of mappers.

REFERENCES

[1] T. Gunarathne, W. Tak-Lon, J. Qiu, G. Fox, "MapReduce in the clouds for science," 2nd IEEE Conference on Cloud Computing Technology and Science, CloudCom-2010, pp. 565–572, 2010.

[2] R. Ananthanarayanan, K. Gupta, P. Pandey, H. Pucha, P. Sarkar, M. Shah, R. Tewari, "Cloud analytics: Do we really need to reinvent the storage stack?," In Proc. of the HotCloud Workshop, San Diego, 2009.

[3] R. L. Grossman, "The Case for Cloud Computing," IT Professional, vol.11, no.2, pp.23-27, March-April 2009.

[4] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, 2004.

[5] K. Shvachko, H. Huang, S. Radia, R. Chansler, "The hadoop distributed file system," in Proc. of the 26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies, 2010.

[6] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," In Proc. of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 (EuroSys '07), 2007.

[7] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," In USENIX OSDI, 2008.

[8] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, E. Harris, "Reining in the outliers in map-reduce clusters using Mantri," In Proc. of the 9th USENIX OSDI Symposium, 2010.

[9] K. Ren, Y. Kwon, M. Balazinska, B. Howe, "Hadoop's adolescence: a comparative workload analysis from three research clusters," Tech. Report UW-CSE-12-06-01, University of Washington, 2012.

[10] Y. Chen, S. Alspaugh, R. H. Katz, "Design insights for MapReduce from diverse production workloads," Technical Report UCB/EECS-2012-17, EECS Dep., University of California, Berkeley, 2012.

[11] G. Ananthanarayanan, A. Ghodsi, S. Shenker, I. Stoica, "Effective straggler mitigation: attack of the clones," In Proc. of the 10th Symp. on Networked Systems Design and Implementation (NSDI), 2013.

[12] F. Ahmad, S. T. Chakradhar, A. Raghunathan, T. N. Vijaykumar, "Tarazu: optimizing mapreduce on heterogeneous clusters," In Proc. of the 17th ASPLOS Conf., 2012.

[13] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, E. Harris, "Scarlett: Coping with Skewed Popularity Content in MapReduce Clusters," In ACM EuroSys, 2011.

[14] H. Karloff, S. Suri, S. Vassilvitskii, "A model of computation for MapReduce," In Proc. ACM-SIAM Sympos. Discrete Algorithms (SODA), pp. 938–948, 2010.

[15] B. Li, E. Mazur, Y. Diao, A. McGregor, P. Shenoy, "A Platform for Scalable One-Pass Analytics using MapReduce," In Proceedings of ACM SIGMOD Conf., 2011.

[16] X. Yang, J. Sun, "An analytical performance model of mapreduce," In Proc. of Cloud Computing and Intelligence Systems (CCIS), 2011.

[17] X. Lin, Z. Meng, C. Xu, M. Wang, "A practical performance model for hadoop mapreduce," In Proc. Of CLUSTER Workshops, 2012.

[18] E. Krevat, T. Shiran, E. Anderson, J. Tucek, J.J. Wylie, G.R. Ganger, "Applying Performance Models to Understand Data-intensive Computing Efficiency," Technical Report CMU-PDL-10-108, Carnegie Mellon University, Pittsburgh, 2010.

[19] Y. Kwon, M. Balazinska, B. Howe, J. Rolia, "SkewTune: Mitigating skew in MapReduce applications," In Proc. of SIGMOD Conf., pages 25–36, 2012.

[20] J. Tan, X. Meng, L. Zhang, "Delay tails in MapReduce scheduling," in Proc. of the SIGMETRICS/PERFORMANCE Conf., 2012.

[21] J. Tan, Y. Wang, W. Yu, and L. Zhang, "Non-work-conserving effects in MapReduce: diffusion limit and criticality," In Proc. of the 14th ACM SIGMETRICS, Austin, Texas, USA, 2014.

[22] M. Lin, J. Tan, A. Wierman, L. Zhang, "Joint optimization of overlapping phases in MapReduce," Performance Evaluation, 2013.

[23] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmleegy, and R. Sears, "Mapreduce online," In Proc. of NSDI, 2010.

[24] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, J. Zhou, "Scope: Easy and efficient parallel processing of massive data sets," In Proc. of VLDB, 2008.

[25] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, "Hive - a warehousing solution over a Map-Reduce framework," PVLDB, vol.2, no.2, pp. 1626–1629, 2009.

[26] S. Li, S. Hu, S. Wang, L. Su, T. Abdelzaher, I. Gupta, R. Pace. "WOHA: Deadline-Aware Map-Reduce Workflow Scheduling Framework over Hadoop Cluster," In Proc. of 34th International Conference on Distributed Computing Systems (ICDCS), 2014.

[27] V.A. Saletore, K. Krishnan, V. Viswanathan, M.E. Tolentino, "HcBench: Methodology, Development, and Characterization of a Customer Usage Representative Big Data/Hadoop Benchmark," IEEE International Symposium on Workload Characterization, 2013.

[28] M. A. Marsan, G. Chiola, "On Petri Nets with Deterministic and Exponentially Distributed Firing Times," Advances in Petri Nets, LNCS, vol. 266, Springer, pp. 132-145, 1987.

[29] A. Feldmann, W. Whitt, "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models", In Proc. of IEEE INFOCOM, 1997.

[30] J. Li , Y.S. Fan, M.C. Zhou "Performance modeling and analysis of workflow," IEEE Trans. on Systems, Man, Cybernetics—Part A: Systems and Humans, vol. 34, no. 2, pp. 229-242, 2004.

[31] R.B.J.T. Allenby and A.B. Slomson, "How to count: An introduction to combinatorics," 2nd ed. CRC Press, pp. 51-60, 2011.

[32] B. Kemper, M. Mandjes, "Mean sojourn times in two-queue fork-join systems: bounds and approximations," OR Spectrum 34(3), 2012.

[33] A.S. Lebrecht, W.J. Knottenbelt, "Response Time Approximations in Fork-Join Queues," in Proc. of the 23rd UK Performance Engineering Workshop (UKPEW), Edge Hill, UK, July, 2007.

[34] H. Schwetman, "CSIM: A C-based, process oriented simulation language," In Proceedings of the 1986 Winter Simulation Conference, pp. 387–396, December 1986.

[35] A. Kaw, E. Kalu, "Numerical Methods with Applications," Holistic Numerical Methods Institute, Dec. 2008.

[36] A. Makowski, S. Varma, "Interpolation approximations for symmetric fork-join queues," Performance Evaluation, vol.20, 145-165, 1994.

[37] V. J. Reddi, B. Lee, T. Chilimbi, K. Vaid, "Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency," in Proc. of ISCA, 2010.

[38] D. Gross, J. F. Shortle, J. M. Thompson, C. M. Harris, "Fundamentals of Queueing Theory," 4th ed. John Wiley&Sons, 2008.

Appendix

A: Proof of Lemma 1

The random variable of waiting time (T_W) can be written as the summation of partial waiting times of the requests in the queue ($Q_1, Q_2, \dots, Q_n$). Then we have:

$$T_{W_n} = Q_1 + Q_2 + \cdots + Q_n$$

where Q_i's distribution is DED $i \in \{1, \dots, n\}$, i.e.:

$$P(Q_i = t) = \mu e^{-\mu(t-T)} U(t-T)$$

where $T = 1/C\mu$ is very small. Then by induction it can be shown that T_W has a delayed Erlang-n distribution, i.e.:

$$P_n = P(T_{W_n} = t) = \mu \frac{(\mu(t-nT))^{n-1}}{(n-1)!} e^{-\mu(t-nT)} U(t-nT)$$

The steady-state arrival point distribution can be written [10]:

$$q_n = (1-r_0){r_0}^n$$

Now we can find the distribution of waiting time by getting the expected value of $q_n P_n$ as follows:

$$P(T_W = t) = f_W(t) = E_n\{q_n P_n\}$$
$$= E_n\left\{(1-r_0){r_0}^n \mu \frac{(\mu(t-nT))^{n-1}}{(n-1)!} e^{-\mu(t-nT)} U(t-nT)\right\}$$
$$= (1-r_0) r_0 \mu e^{-\mu t} E_n\left\{\frac{(\mu r_0 e^{\mu T}(t-nT))^{n-1}}{(n-1)!} e^{\mu T} U(t-nT)\right\}$$

Let $P(T_W = t) = 0$ for $t < Dr_0 \approx mT$ where D is the deterministic coefficient to refine the relation. Using $t \gg T$ and Taylor's series approximation of $e^{\mu T} \approx 1 + \mu T$ or we can say that nT is not stochastic and it is deterministically equal with Dr_0, because the average amount of time needed to wait for job arrived with the $mean = \lambda$ is $D\lambda/\mu = Dr_0$. We have:

$$P(T_W = t | t > Dr_0)$$
$$= (1-r_0) r_0 \mu e^{-\mu t} \sum_{n=0}^{t/T} \frac{(\mu r_0 (t - E\{nT\}))^{n-1}}{(n-1)!} e^{\mu E\{nT\}}$$
$$\approx (1-r_0) r_0 \mu e^{-\mu t} e^{\mu r_0 (t - Dr_0 + D)} = (1-r_0) r_0 \mu e^{-\mu(1-r_0)(t-Dr_0)}$$

Then, similar to the G/M/1 deduction for response time, we can write the response time distribution as follows:

$$P(T_R = t) = f_R(t) = (1-r_0)\mu e^{-\mu(1-r_0)(t-Dr_0)} U(t-Dr_0)\,. \quad \square$$

B: Proof of Lemma 4

Using the method of Lagrange multipliers, it can be inferred that the set of non-linear equations optimizing the number of mapper nodes is separable from the set of non-linear equations optimizing their task arrival rates. We have:

$$\min_{N_m, \underline{\lambda}} (MST) = \min_{N_m, \underline{\lambda}} \int_0^\infty t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m} F_{R_{M_i}}(t)\right) dt$$
$$S.t.\ \lambda = \sum_{i=1}^{N_m} \lambda_i\,;\ \underline{\lambda} = \left[\lambda_1 \lambda_2 \lambda_3\ \dots \lambda_{N_m}\right]^T$$
$$\nabla_{N_m, \underline{\lambda}, \alpha}\left(MST - \alpha\left(\lambda - \sum_{i=1}^{N_m} \lambda_i\right)\right) = 0$$
$$\left.\begin{aligned} \frac{\partial MST}{\partial \lambda_1} = \frac{\partial MST}{\partial \lambda_2} = \cdots = \frac{\partial MST}{\partial \lambda_{N_m}} = \alpha \\ \frac{\partial MST}{\partial N_m} = 0 \end{aligned}\right\} \rightarrow Optimal\ "N_m\ and\ \underline{\lambda}"$$

Similarly for the optimization problem $\min_{N_m}\left(\min_{\underline{\lambda}}(MST)\right)$ we will reach the same non-linear equations, which give the optimal solution. So these two optimization problems are separable. $\square$

C: Proof of Lemma 6

The CDF of the response time of i^{th} mapper node as a function of its service rate μ_i and arrival rate λ_i is $\left(1 - e^{-(\mu_i - \lambda_i)t}\right)$. Then MST will be:

$$MST = \int_0^\infty t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m}\left(1 - e^{-(\mu_i - \lambda_i)t}\right)\right) dt$$

Based on the equilibrium property (Def.4) we have:

$$\forall i, j \in \{1,2,\dots,N_m\}: \frac{\partial MST}{\partial \lambda_i} = \frac{\partial MST}{\partial \lambda_j}$$
$$\leftrightarrow \int_0^\infty t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m} \frac{\partial}{\partial \lambda_i}\left(1 - e^{-(\mu_i - \lambda_i)t}\right)\right) dt$$
$$= \int_0^\infty t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m} \frac{\partial}{\partial \lambda_j}\left(1 - e^{-(\mu_j - \lambda_j)t}\right)\right) dt$$
$$\leftarrow \frac{\partial}{\partial \lambda_i}\left(1 - e^{-(\mu_i - \lambda_i)t}\right) = \frac{\partial}{\partial \lambda_j}\left(1 - e^{-(\mu_i - \lambda_i)t}\right)$$
$$\leftarrow t e^{-(\mu_i - \lambda_i)t} = t e^{-(\mu_j - \lambda_j)t} \leftarrow \mu_i - \lambda_i = \mu_j - \lambda_j$$

Now we have:

$$\mu_1 - \lambda_1 = \mu_2 - \lambda_2 = \cdots = \mu_{N_m} - \lambda_{N_m} = \left(\sum_{j=1}^{N_m} \mu_j - \lambda\right)/N_m$$

Then we have:

$$\lambda_i = \mu_i - \frac{\sum_{j=1}^{N_m} \mu_j - \lambda}{N_m}$$

Having $\lambda_i > 0$; $\forall i$ then we have:

$$min(\mu_1, \mu_2, \dots, \mu_{N_m}) = \mu_{min} > \frac{\sum_{j=1}^{N_m} \mu_j - \lambda}{N_m}. \quad \square$$

D: Proof of Lemma 7

The CDF of response time of the i^{th} mapper as a function of $A_i(t)$ (inter-arrival time CDF) and service rate μ_i is $1 - e^{-\mu_i(1-r_{0_i})t}$ where r_{0_i} is the unique real root of $z = A_i^*[\mu_i(1-z)]$ in (0,1) and A_i^* is the LST (Laplace–Stieltjes transform) of $A_i(t)$ [38]. Based on Lemma 8, Equation (28) and similarity to M/M/1 form (Lemma 6), we have:

$$\forall i, j \in \{1,2,\dots,N_m\}: F_{R_{M_i}}(t) = F_{R_{M_j}}(t)$$
$$\leftarrow 1 - e^{-\mu_i(1-r_{0_i})t} = 1 - e^{-\mu_j(1-r_{0_j})t}$$
$$\leftarrow \mu_i(1-r_{0_i}) = \mu_j(1-r_{0_j}). \quad \square$$

E: Proof of Lemma 8

From (14) we have:

$$MST = \int_0^\infty t \frac{\partial}{\partial t}\left(\prod_{i=1}^{N_m} F_{R_{M_i}}(t)\right) dt$$

From (16) of Def.4 we have the following equilibrium property:

$$\forall i, j \in \{1,2,\dots,N_m\}: \frac{\partial MST}{\partial \lambda_i} = \frac{\partial MST}{\partial \lambda_j}$$
$$\leftrightarrow \int_0^\infty t \frac{\partial}{\partial t}\left(F_{R_{M_1}}(t) F_{R_{M_2}}(t) \dots F_{R_{M_{i-1}}}(t)\left[\frac{\partial}{\partial \lambda_i} F_{R_{M_i}}(t)\right] F_{R_{M_{i+1}}}(t) \dots F_{R_{M_{N_m}}}(t)\right) dt$$
$$= \int_0^\infty t \frac{\partial}{\partial t}\left(F_{R_{M_1}}(t) F_{R_{M_2}}(t) \dots F_{R_{M_{j-1}}}(t)\left[\frac{\partial}{\partial \lambda_j} F_{R_{M_j}}(t)\right] F_{R_{M_{j+1}}}(t) \dots F_{R_{M_{N_m}}}(t)\right) dt$$

$$\leftrightarrow \int_0^\infty t\frac{\partial}{\partial t}\left(\frac{1}{F_{R_{M_i}}}\frac{\partial}{\partial \lambda_i}F_{R_{M_i}}(t)\prod_{k=1}^{N_m}F_{R_{M_k}}(t)\right)dt$$
$$= \int_0^\infty t\frac{\partial}{\partial t}\left(\frac{1}{F_{R_{M_j}}}\frac{\partial}{\partial \lambda_j}F_{R_{M_j}}(t)\prod_{k=1}^{N_m}F_{R_{M_k}}(t)\right)dt$$
$$\leftarrow \frac{1}{F_{R_{M_i}}}\frac{\partial}{\partial \lambda_i}F_{R_{M_i}}(t) = \frac{1}{F_{R_{M_j}}}\frac{\partial}{\partial \lambda_j}F_{R_{M_j}}(t)$$

So it can be inferred that (28) is a sufficient condition to have an optimal solution. For the second condition (29) we will show that any deviation from the distributions' equality may increase MST. The proof is dedicated to two different distributions that can be easily generalize to any number by induction, so we have:

$$MST = \int_0^\infty t\frac{\partial}{\partial t}\left(\prod_{i=1}^{2}F_i(t)\right)dt = \int_0^\infty t\frac{\partial}{\partial t}(F_1F_2)dt$$

Any ε-deviation from the distributions equilibrium can be written as follows:

$$\lambda_1 + \lambda_2 = (\lambda_1 + \varepsilon) + (\lambda_2 - \varepsilon) = \lambda$$
$$F(t) = F_1(t,\lambda_1) = F_2(t,\lambda_2)\ or\ F = F_1(\lambda_1) = F_2(\lambda_2)$$
$$\frac{d}{dt}F(t) = f(t) = f_1(t,\lambda_1) = f_2(t,\lambda_2)\ or\ f = f_1(\lambda_1) = f_2(\lambda_2)$$

We have to show that:

$$MST(\lambda_1,\lambda_2) = \int_0^\infty t(f_1(\lambda_1)F_2(\lambda_2) + f_2(\lambda_2)F_1(\lambda_1))dt$$
$$\leq MST(\lambda_1+\varepsilon,\lambda_2-\varepsilon) = \int_0^\infty t(f_1(\lambda_1+\varepsilon)F_2(\lambda_2-\varepsilon) + f_2(\lambda_2-\varepsilon)F_1(\lambda_1+\varepsilon))dt$$

Equivalently:

$$\Delta MST = MST(\lambda_1+\varepsilon,\lambda_2-\varepsilon) - MST(\lambda_1,\lambda_2)$$
$$= \int_0^\infty t(f_1(\lambda_1+\varepsilon)F_2(\lambda_2-\varepsilon) + f_2(\lambda_2-\varepsilon)F_1(\lambda_1+\varepsilon))dt - 2\int_0^\infty tfFdt$$
$$\geq 0$$

We can write Taylor's series of the new deviated distributions as:

$$F_1(t,\lambda_1+\varepsilon) = F + \frac{F_1'(\lambda_1)}{1!}\varepsilon + \frac{F_1''(\lambda_1)}{2!}\varepsilon^2 + \cdots = F + \varepsilon F_1'(\xi_1)$$

$$F_2(t,\lambda_2-\varepsilon) = F - \frac{F_2'(\lambda_2)}{1!}\varepsilon + \frac{F_2''(\lambda_2)}{2!}\varepsilon^2 + \cdots = F - \varepsilon F_2'(\xi_2)$$

where $\exists\xi_1,\xi_2$: $\lambda_1 \leq \xi_1 \leq \lambda_1+\varepsilon$, $\lambda_2 \leq \xi_2 \leq \lambda_2+\varepsilon$ and second equalities come from the *Mean Value Theorem*. Now the difference of two above distribution is as follows:

$$\Delta\text{MST} = \int_0^\infty t\Big(\big(f+\varepsilon f_1'(\xi_1)\big)\big(F-\varepsilon F_2'(\xi_2)\big) + \big(f-\varepsilon f_2'(\xi_2)\big)\big(F+\varepsilon F_1'(\xi_1)\big)\Big)dt - 2\int_0^\infty tfFdt$$

Approximating the difference by ignoring higher order ε, we have:

$$\Delta MST \cong \int_0^\infty t(-\varepsilon fF_2' + \varepsilon f_1'F + \varepsilon fF_1' - \varepsilon f_2'F)dt$$
$$= \varepsilon\frac{d}{d\lambda}\left(\int_0^\infty tfFdt\right)_{\xi_2}^{\xi_1}$$

We know that MST is a monotonically increasing function with respect to λ as MST is a notion of aggregated response time. Consequently the difference value is non-negative. Also (30) and (31) are other representations of the CDF (29), because the equality of moment-generating functions is equivalent with the equality of the LST of distributions. □

F: Proof of Lemma 10

Given random variable X, convex function $\Phi(X)$, and expected value operation $E\{.\}$ the Jensen inequality states that:

$$\Phi(E\{X\}) \leq E\{\Phi(X)\}$$

We know that max(X) is a convex function, from the above inequality and (7) we have:

$$MST = E\left\{max\left(R_{M_1}, R_{M_2}, \dots, R_{M_{N_m}}\right)\right\} \geq max\left(E\{R_{M_1}\}, E\{R_{M_2}\}, \dots, E\left\{R_{M_{N_m}}\right\}\right)$$

which shows the lower bound of MST. The minimum value of the lower bound is achieved when the mean response times are equal, because:

$$\min\left(max\left(E\{R_{M_1}\}, E\{R_{M_2}\}, \dots, E\left\{R_{M_{N_m}}\right\}\right)\right) = E\{R_{M_i}\};\forall i$$

Equivalently it is similar to a mini-max strategy in game theory when the elements are continuously bounded, and the solution happens when all elements are equal. Or it can be assumed that there are $p_1, p_2, \dots, p_{N_m}$ probabilities where $\sum_{i=1}^{N_m} p_{N_m} = 1$ and:

$$max\left(E\{R_{M_1}\}, E\{R_{M_2}\}, \dots, E\left\{R_{M_{N_m}}\right\}\right) = p_1E\{R_{M_1}\} + p_2E\{R_{M_2}\} + \cdots + p_{N_m}E\left\{R_{M_{N_m}}\right\}$$

Using Lagrange Multipliers method the minimum of the above cost function as lower bound occurs when the mean response times are the same as (37). For the upper bound from (9) we have:

$$MST = \sum_{j=1}^{N_m}\int_0^\infty t\prod_{i=1}^{N_m}F_{R_{M_i}}(t)\frac{f_{R_{M_j}}(t)}{F_{R_{M_j}}(t)}dt = \sum_{j=1}^{N_m}\int_0^\infty t\frac{\prod_{i=1}^{N_m}F_{R_{M_i}}(t)}{F_{R_{M_j}}(t)}f_{R_{M_j}}(t)dt = \sum_{j=1}^{N_m}\int_0^\infty tP_j(t)f_{R_{M_j}}(t)dt$$

where the multiplicative function is less than one because:

$$P_j(t) = \frac{\prod_{i=1}^{N_m}F_{R_{M_i}}(t)}{F_{R_{M_j}}(t)} = F_{R_{M_1}}(t)F_{R_{M_2}}(t)\dots F_{R_{M_{j-1}}}(t)F_{R_{M_{j+1}}}(t)\dots F_{R_{M_{N_m}}}(t) \leq 1$$

So we have:

$$MST = \sum_{j=1}^{N_m}\int_0^\infty tP_j(t)f_{R_{M_j}}(t)dt \leq \sum_{j=1}^{N_m}\int_0^\infty tf_{R_{M_j}}(t)dt = \sum_{i=1}^{N_m}E\{R_{M_i}\}$$

which shows the upper bound of MST. Using the method of Lagrange multipliers, the minimum of the upper bound happens when:

$$\frac{d}{d\lambda_i}E\{R_{M_i}\} - \alpha = 0;\ \forall i$$

This is equivalent to (38). □