GDSim: Benchmarking Geo-Distributed Data Center Schedulers

Daniel Alves
University of California, Santa Cruz
USA
dalves@ucsc.edu

Katia Obraczka
University of California, Santa Cruz
USA
katia@soe.ucsc.edu

Abdul Kabbani
Facebook
USA
akk@fb.com

***Abstract*—As cloud providers scale up their data centers and distribute them around the world to meet demand, proposing new job schedulers that take into account data center geographical distribution have been receiving considerable attention from the data center management research and practitioner community. However, testing and benchmarking new schedulers for geo-distributed data centers is complicated by the lack of a common, easily extensible experimental platform. To address this gap, we propose GDSim, an open-source job scheduling simulation environment for geo-distributed data centers that aims at facilitating development, testing, and evaluation of new geo-distributed schedulers. We showcase GDSim by using it to reproduce experiments and results for recently proposed geo-distributed job schedulers, as well as testing those schedulers under new conditions which can reveal trends that have not been previously uncovered.**

***Index Terms*—data center, job scheduling, geo-distributed**

I. Introduction

Geographic redundancy and physical diversity is an important technique used by cloud computing services. By distributing their data centers around the world, companies such as Google and Amazon, can not only minimize the chance that failures will not keep their customers from accessing important data, but also can reduce normative user-perceived access time. For example, Google has more than 20 data centers across four continents [1], Microsoft's lists more than 50 data centers [2], and Facebook's expansion has resulted in over 15 data centers world-wide [3]. But geographic redundancy is not a cure-all and presents its own problems. The physical distance separating data centers, coupled with resource limitations of the network connecting them, make scheduling algorithms that perform well for centralized data centers, inefficient for their geo-distributed counterparts. For example, one well-known algorithm, Shortest Remaining Processing Time, ignores the extra cost of transferring information (jobs and data) over the network [4]. As such, job scheduling for geo-distributed data centers has recently emerged as an active topic of research.

A number of job schedulers for geo-distributed data centers have been proposed, but the research and practitioner communities are still trying to address the question of how to evaluate new geo-distributed schedulers and how to compare them to existing ones for different data center scenarios, applications, and workloads. Currently, most existing geo-distributed schedulers use their own testing environment, e.g., custom simulators or sometimes in the form of testbeds that emulate geo-distributed data centers using geographically distributed servers hosted by cloud providers such as AWS. This status-quo makes the task of comparing different schedulers in a systematic, reproducible, and scientifically sound manner either unfeasible or very complicated and cumbersome.

Many disciplines that rely on experimental research emphasize the importance of providing open-source platforms for evaluating new systems. This is notably the case in computer architecture [5], [6] and networking [7], [8], for example. In the specific case of data center job scheduling, few simulators that can be used for comparative evaluation studies have been proposed–but they only apply to centralized data center environments [9]–[13], and thus cannot be used in the context of geo-distributed data center scheduling with their current capabilities. This paper presents a simulation platform that, by specifically targeting job schedulers for geo-distributed data centers, fills this gap. The proposed platform, GDSim [14], aims to offer an open-source benchmarking environment available to researchers and practitioners that allows side-by-side comparative studies of different schedulers under a variety of scenarios and conditions. By serving as a level playing field to test and evaluate geo-distributed data center schedulers, GDSim will also help advance the state-of-the-art in geo-distributed data centers by facilitating the development of new schedulers as data center architecture, usage, and applications evolve. GDSim can complement testbed-based experimental evaluation as a low-cost alternative that provides controlled and reproducible experiments under a wide range of scenarios.

GDSim consists of two main functional components: a simulator and a workload generator. The workload generator extends the work of Chen et al. [15] on synthetic workload generation to generate a wide range of workloads to help address the limited availability of public data center workloads. In addition to real workload traces, GDSim simulations can be driven by its own generated workloads based on different features (e.g., number of jobs, number of tasks per jobs, etc) and various geo-distributed data center topologies to evaluate different job schedulers. GDSim can also be easily extended to include various job scheduling algorithms.

This work was partially supported by CAPES (Coordination for the Improvement of Higher Education Personnel - Brazil) and by the US National Science Foundation under project CNS 1321151.

In this paper, we showcase GDSim by using it to reproduce results from recently proposed geo-distributed data center schedulers. In our experiments, we demonstrate GDSim's ability to generate synthetic workloads, mimicking real production workloads. We also show that by exposing schedulers to a wide variety of workloads, GDSim can reveal trends not previously identified.

The rest of this paper is organized as follows: Section II describes the job scheduling problem in geo-distributed data centers and related work in benchmarking data center schedulers; next, we present GDSim in Section III; Section IV describes our methodology as we showcase how GDSim can be used to evaluate geo-distributed data center schedulers; our results and observations are discussed in Section V; Section VI concludes the paper with directions for future work.

II. Job Scheduling in Geo-Distributed Data Centers and Related Work

Job scheduling is the problem of building the ideal execution schedule for a set of jobs given a limited set of resources. It is particularly relevant when discussing efficient usage of computing clusters and data centers [16]–[19]. In addition to minimizing job completion times, another important consideration is reducing operational costs, especially as data centers scale to very large numbers of servers, as evidenced by recent research on reducing data center energy costs [18].

The job scheduling problem can be formulated in many different ways; however, some elements are common: there is always a set of *jobs* that need to be scheduled, each of which has one or more *tasks*. Tasks may have each different expected *durations* and require a set amount of computing *resources* and access to specific *files*. For simplicity, it is common to assume that all tasks of the same job have the same resource requirements. While tasks in a job can also have internal dependencies, the geo-distributed job schedulers considered in this paper handle only batches of tasks that can be scheduled concurrently. If a job has internal dependencies, it can be broken into multiple jobs, which are then scheduled as dependencies are resolved. Jobs can either arrive all together, i.e., in batches, or continuously over time. In the context of data center scheduling, it is often more realistic to handle jobs continuously as they arrive.

As previously noted, the problem of job scheduling in geo-distributed data center has recently captured the attention of systems and networking researchers and practitioners. Geo-distributed data centers are data centers that are spread across long distances. While jobs could be scheduled at any of these data centers, their data requirements make the decision of where to schedule jobs more complicated. One approach is to replicate data across multiple locations, but it is infeasible and/or impractical to have replicas at all data centers. Failing to take networking latency into consideration could result in considerably longer data transfer times before a job can be executed. This means that, unlike their centrally located counterparts, geo-distributed data center environments require job schedulers to also account for data transfer times before execution.

While geo-distributed data centers can use schedulers dedicated to specific applications such as scientific computing workloads [20], machine learning [21], and data analytics [22], in this work we are interested in schedulers intended for general use workloads. Workload-Aware Geo-distributed Scheduling (SWAG) [23] is one of the earliest algorithms proposed for job scheduling in geo-distributed data centers. SWAG builds on the Shortest Remaining Processing Time (SRPT) heuristic, but restricts the placement of jobs to locations that already have the data required to execute the job. The reasoning for this is that, while SRPT offers optimal completion times for a single queue and single server scenario [24], it fails to perform optimally with multiple servers [4]. Another early alternative was Flutter [25], which proposed a scheduler that can yield an optimal solution using linear programming. GeoDis [26] is a more recent proposal which adapts SRPT for use in geo-distributed scenarios by computing processing times that include data transfer times accounting for data center occupation at time of scheduling.

A. Performance Benchmarking

A critical step in developing new systems is the evaluation of their performance against a baseline and the state-of-the-art. In the specific case of evaluating data center job schedulers, we need two main components: an environment in which we can implement the scheduler, and a way to generate workloads to drive scheduler execution.

To our knowledge, there are no open-source, publicly available platforms that can be used for testing and validating geo-distributed data center job schedulers. There are already established simulators for non-geo-distributed scenarios, such as the simulator used in the evaluation of Google's Omega [27], or the Yarn Scheduler Load Simulator (SLS) [28]. These simulators have important roles in validating proposals for centralized data centers, however none of them was designed with geo-distributed scheduling in mind. Extending these existing simulators to support geo-distribution would require substantial modifications to their basic structure, which motivated us to create GDSim, an experimental platform that is specifically designed to benchmark job schedulers for geo-distributed data center environments.

Existing schedulers, including the ones considered in our study, have been evaluated either using custom simulators, or on relatively small geo-distributed data center testbeds (e.g., through services such as AWS [29].) While testing on real testbeds provides more realistic results, it is usually a more costly option that is harder to scale, limits the range of conditions under which the system can be tested/validated, and does not provide an experimental environment that is conducive to reproducibility.

B. Workloads

Evaluating schedulers requires comparing their performance under production-like conditions. Since researchers often do

not have access to production data centers, they usually do not have full knowledge of their workloads. Instead, they rely on workload traces that have been made publicly available, some of which by large computing companies for research purposes. However, there are only a few of those traces available and usually they include limited information as described below.

In our work, in addition to using real, publicly available workloads, we also use synthetic workloads generated by GDSim's workload generator. The real workloads we use in our experiments to evaluate geo-distributed job schedulers include traces published by Facebook [15], Google [30], [31], and Alibaba [32]. The first two traces were selected because they were used in previous studies that evaluated the schedulers considered here and thus allow us to validate results obtained by GDSim. The third trace was chosen to illustrate that GDSim is trace-agnostic, i.e., it can be driven by different types of traces, including synthetic workloads. While there are other traces available, our goal here is to showcase how GDSim can be used to study the performance of different geo-distributed data center job schedulers, rather than conducting a performance study comparing different schedulers.

Each of the three real workload traces provides information collected from job executions in computing clusters of the corresponding companies after anonymization. They are provided in different formats and with different amounts of information, so we have to make some adjustments to be able to use them. For our use we want to extract from the traces a list of submitted jobs, each job containing submission time, computing resource requirements, required data (including size and initial location, which might change during execution), number of tasks and expected duration of each task. Most notably, the Facebook trace was synthetically generated based on real Facebook data, which is not publicly accessible nevertheless. The trace also lacks task information, which we ended up extrapolating based on distributions presented in other works [23], [26]. None of the traces included full data file information either, e.g., size of the required data file or even the file name.

The information GDSim uses to schedule consists of the following: job arrival (submission) times, computational (CPU core) requirements, size of required data, data location, number of tasks per job, task execution times, and jobs to data mapping (i.e. the constraint that several jobs require access and contend for the same data). The Facebook trace included file name information, but not file size information, which is also absent from the other traces. For the purpose of using these traces we estimated file size from the input transfer size information, and when file names were absent we used user names as a proxy. None of the traces included information about data locations that could provide a baseline on how data is distributed in those data centers. We should also note that the Google and Alibaba traces were provided as task event traces, that is, the traces describe events related to task such as task submission, launching, or completion. Each event also has related resource usage information, from which we extract CPU and file size requirements. That means we had to reconstruct job descriptions from the events, so the resulting jobs might not be completely accurate. The results are however good enough for the purpose of comparing different schedulers, as we can test with different traces and observe how their differences are reflected in the resulting performance of the schedulers.

Although there are now more publicly available workload traces, their number and variety is still relatively small, and, as a result, they may not be able to subject geo-distributed job schedulers to sufficient variation in conditions such as job arrival (submission) times, computational (CPU core) requirements, size of required data, data location, number of tasks per job, task execution times, and jobs to data mapping (i.e. the constraint that several jobs require access and contend for the same data). For this reason, we looked into approaches to generate synthetic workloads, e.g., the work described in [15] which proposes synthetic workload generation by sampling of jobs from existing workloads. Their approach would, however, preserve existing patterns of behavior, which was their original intention. In our work, we wanted to explore more diverse patterns that are still based on real data, so we developed our own method to generate synthetic traces based on real workload data, which is described in Section III-A.

III. GDSim

In this section, we start by providing an overview of GDSim and then describe its components, namely its workload generator and simulation engine in detail. GDSim aims at providing an experimental platform to allow first-level benchmarking of geo-distributed data center job schedulers. It also serves as a platform to run controlled, reproducible experiments, complementing real distributed data center testbed experimentation. GDSim's current features allow us to reproduce experiments comparing the performance of existing geo-distributed job schedulers, while GDSim's ongoing and future development will aim at extending its features to study dependency dynamics between jobs and the underlying network.

GDSim's workload generator builds synthetic workload traces reproducing behavior seen in real workloads and beyond. As described in detail in Section III-A, by providing "knobs" that allow varying workload features such as number of jobs, number of tasks per job, etc., the workload generator can produce workloads that will subject schedulers to wide range of conditions that may not be possible with existing traces. It uses an *extractor* that reads existing traces and collects information that is used to generate synthetic workloads. Recall that GDSim experiments can also be driven by real traces as shown in Sections IV and V.

The goal of GDSim's simulator is to drive the execution of job schedulers being evaluated using as input real or synthetic workloads and data center configuration/topology definition. It then generates a job schedule based on the rules of the scheduler being studied.

A. Workload Generator

Our synthetic workload generation uses *attribute sampling*, i.e., it considers the main attributes of a workload trace and

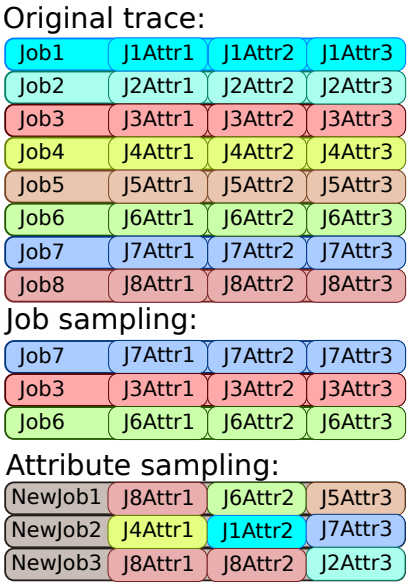

Fig. 1. Workload generation via *job sampling* versus *attribute sampling*.

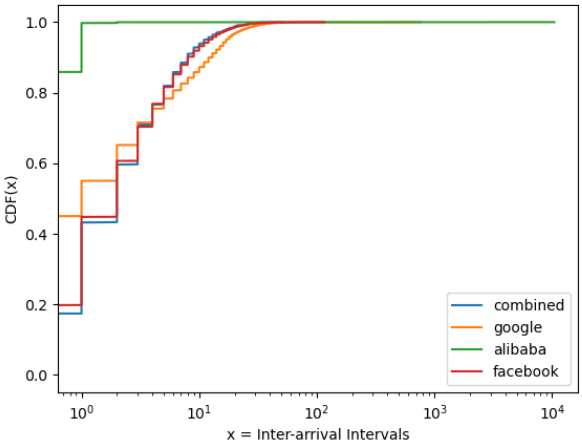

Fig. 2. Job inter-arrival time distribution in original traces and new synthetic trace showing that the synthetic trace's job inter-arrival time behavior is closer to that of the Facebook trace.

sample from them individually, instead of sampling the entire job set (or *job sampling*). While this means that we do not have the same job profiles as the original data, it allows for greater variation of behavior, as well as combining trends from different traces. As data centers and their applications evolve, our goal is to expose scheduling algorithms under evaluation to a wider diversity of scenarios, beyond what existing traces may present, but preserving existing tendencies instead of using completely random data.

Fig. 1 illustrates how our workload generation approach differs from [15]. When a synthetic trace is created through job sampling, it copies entire jobs from the original trace. This preserves relations between attributes even if they had not been explicitly identified. For example, it is possible that in a given trace all jobs with few tasks have long lasting tasks, and that relationship would be preserved. Attribute sampling, on the other hand, creates jobs by sampling each attribute from previously observed values. This preserves individual distributions for each attribute, but may lose relationships between attributes, such as the correlation between number of tasks and their duration. Nevertheless, this allows us to combine patterns observed in different traces, so for example if one trace has all its jobs made up of few long tasks, and another has jobs made up of many short tasks, we can create traces with jobs consisting of many long tasks and traces with few short tasks.

GDSim's workload generator works as follows: first, an extractor reads real workload traces whose characteristics we want to reproduce, then the generator creates a predefined number of jobs through sampling the desired characteristics of each workload. The generator can also be configured to use traditional statistical distributions, such as Pareto or Zipf, if this is more suitable for certain workloads.

Combining Workloads: To demonstrate how the workload generator allows us to create synthetic traces that combine characteristics of different workloads, we created a synthetic trace of one thousand jobs that reproduces the interarrival times of the Facebook trace, the number of tasks and task durations of the Alibaba trace, and is similar to the Google trace in the remaining characteristics. As an example, Figure 2 shows how the synthetic trace reproduces the statistical behavior of the Facebook trace in regard to interarrival times.

B. Simulator

GDSim's event-driven simulator is responsible for modeling events as they happen, including their actions, and the changes they cause in the simulated system. Its main parameters are the scheduler being simulated and the *scheduler hook*, which determines how the scheduler is called. It takes as input the data center topology/configuration descriptions and the workload in the form of jobs that will arrive and the files they need and where they are stored. It is also responsible for logging job execution information so that later it can report performance metrics.

a) Events: Events in our simulator can be categorized in terms of their dependence on the network. As an example of that, consider the scheduling of a job's computing task. The event that defines the start of that task may depend on network activity if there are files that have to be transferred to the chosen data center before the task can start, but once the task has started, the conclusion of that same task is an event that does not depend on network activity. The purpose of this distinction is to allow us to separate network-dependent events so that the underlying network can be modeled separately. As illustrated in Fig. 3, the simulation loop then consists of identifying the time of the next network-independent event, verifying if any network dependent events happen before that time, and finally evaluating the earliest event to update the

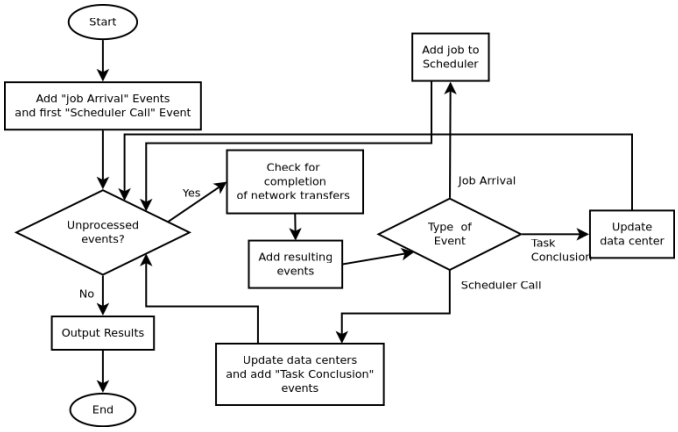

Fig. 3. GDSim's simulator main loop.

state of the simulation. The main network-independent events model job arrivals, scheduler calls, and task conclusions. A job arrival event represents the submission of a job to the scheduler, and results in an update on the scheduler state. A scheduler call event results in the execution of the scheduling algorithm, which in turn updates the state of simulated data centers to represent execution tasks and creates corresponding task conclusion events at the appropriate times. A task conclusion event then causes another update to the state of the simulated data centers, representing the ending of the corresponding execution, and updates the corresponding job execution information.

When it first starts, the simulator creates the state to be tracked from the input, job arrival events for each job present in the workload trace, as well as the first scheduler call event. Future scheduler call events will be created according to the scheduler *hook*, which, in our current implementation can be configured as either *online* or *window-based* mode. In online mode, a scheduler call is set for after the arrival of each group of jobs with the same timestamp, while in window-based mode, each scheduler call happens periodically based on a pre-specified time window for as long as there are jobs to schedule. These different approaches represent a trade-off between responsiveness and scheduler processing overhead. In other words, while an online hook can result in less wait for the job to be scheduled, it will cause more calls to the schedulers; the window-based hook, on the other hand, reduces scheduler calls but can result in wasted resources as the jobs are not running while waiting to be scheduled. We implement these modes, as well as the means to implement other modes, to allow for better exploration of how the schedulers would be used, although we must note that the simulator does not model the execution time of the scheduler itself and that could impact the results if care is not taken when preparing the simulation.

The simulator processes events until there are no more jobs to schedule and all tasks have concluded. Then it outputs the job execution information that can be used to calculate the desired performance metrics. Fig. 3 illustrates GDSim simulator's main loop.

b) Data Center Model: Since the simulator cannot fully reproduce multiple data centers executing real computing jobs, we have to make some assumptions to model geo-distributed data centers.

We assume that the data center topology will stay constant throughout the simulation period, and that the scheduler has access to information describing the state of the data center and its distributed sites at any time. The former is a simplification that is reasonable as infrastructure tends to change little during short windows of time, and even inter-data center bandwidth can be managed through establishment of private WANs [33]. The latter is an important aspect of geo-distributed schedulers and is itself subject of further research [34]–[36].

Internally, data centers are assumed to serve jobs in a First Come First Serve (FCFS) fashion, in line with current literature models [23]. Each data center contains multiple nodes of possibly different configurations. We abstract away possible hardware resource differences between servers in a data center and focus on their maximum capacity using a "slot" to represent the capacity for serving simultaneous tasks, while allowing a task to consume more than one slot for execution, depending on the task's resource requirements.

c) Job Model: When modeling jobs, we assume that all tasks in a job require access to the same data. While this is not realistic, it is also a common assumption and is consistent with the fact that this information is typically absent in publicly available traces. In a similar way we assume that all tasks in a job are independent from each other, all jobs are also independent from each other.

As much as possible, we aimed to keep our assumptions in line with current practices as described in the literature. Firstly, we assume that it is possible to estimate the duration of a submitted task. This is a common assumption that holds well for recurring tasks [37] and we can also justify it by assuming that instead of using the real duration, the scheduler can use the specified deadline in its calculations. The geo-distributed job schedulers considered also assume no preemption, task failures, or rescheduling.

C. Schedulers

GDSim currently includes the implementation of three schedulers: Global-SRPT, a modification of the traditional Shortest Remaining Processing Time algorithm adapted to multiple data centers; SWAG [23], one of the first schedulers for geo-distributed data centers; and GeoDis [26], representing the current state-of-the-art. They were also chosen to validate GDSim based on the performance results presented by Hung, Golubchik and Yu [23] and by Convolbo, Chou, Hsu and Chung [26].

Global-SRPT selects the job with the least remaining processing time and schedules that job's tasks to run on data centers with available computing capacity that have the best expected time to retrieve the data. Workload-Aware Geo-distributed Scheduling (SWAG) uses a similar idea, but restricts placement of tasks to data centers that already have the data tasks need to execute. It does not allow data migration

TABLE I
TOPOLOGY CAPACITY

Data Center	Computers	Cores per Computer
1	13	24
2	7	12
3	7	8
4	12	8
5	31	32
6	31	32
7	10	16
8	8	12

TABLE II
BANDWIDTH (MBPS)

	1	2	3	4	5	6	7	8
1	-	931	376	822	99	677	389	935
2	931	-	97	672	381	82	408	93
3	376	97	-	628	95	136	946	175
4	822	672	628	-	945	52	77	50
5	99	381	95	945	-	822	685	535
6	677	82	136	52	822	-	69	639
7	389	408	946	77	685	69	-	243
8	935	93	175	50	535	639	243	-

between data centers to facilitate scheduling, and instead of using total processing time, it uses expected completion time for the job, given the current status of all data centers, and assuming there are no other jobs to schedule. For each job to be scheduled, SWAG calculates the expected completion time by estimating when the last task in the job would finish, based on an assumption that no other jobs will arrive within the service time. GeoDis extends SWAG by attempting to take advantage of replication and factoring it into the calculation of the scheduling weight of each job and then scheduling jobs in the order of their weights.

IV. EXPERIMENTAL METHODOLOGY

In order to showcase GDSim and how it can be used as a benchmarking platform for geo-distributed data center schedulers, we use run experiments to evaluate the schedulers currently implemented using real- and synthetic workloads and different data center topologies.

One of our goals was to validate GDSim by showing that it can reproduce experimental results similar to those previously observed. To this end, we run experiments comparing GeoDis and SWAG against an adaptation of SRPT for use with geo-distributed data centers.

Next, we experimented with other workloads and topologies to show how GDSim allows us to explore more scenarios, which can reveal behavior not yet observed and lead to new findings and insight on designing efficient geo-distributed schedulers.

We reproduced the topology used in Convolbo, Chou, Hsu and Chung [26] with same number of computers, computing cores per computer and the same bandwidth between data centers. Table I shows the capacity of each data center, and Table II shows the bandwidth between data centers.

We also created a smaller topology to illustrate how the topology itself is an important parameter when evaluating schedulers. The smaller topology has the same number of data centers, but each data center has only one computer and all computers but one have only one core. The last computer has 32 cores to allow larger jobs to run. This topology uses the same inter-data center bandwidth values as shown in Table II.

We also varied how files are distributed across the data centers. To this end, we distributed the files randomly using a Zipf distribution with the following values for the skew parameter: $[1.01, 2, 5, 10]$. The higher the skew parameter is, the more likely files are co-located.

To measure schedule performance, we use two main metrics: schedule makespan and mean job latency. The schedule makespan measures the time from the arrival of the first job to the conclusion of the last job. This is a more traditional job scheduling metric, but is not as representative in the context of scheduling in a data center that is servicing requests from multiple independent sources. The mean job latency, on the other hand, measures average job latency, which is the time difference between the job submission and the completion of its last task, averaged over all jobs. This metric does a better job of capturing the overall effectiveness of the scheduler in terms of its per-job impact.

A. Schedulers

As previously noted, we used three existing geo-distributed job schedulers in our experiments: global-SRPT, SWAG and GeoDis. We selected these schedulers due to their relatively simple implementation as well as the fact that they are well-known and considered the state-of-the-art in the field, which allows us to validate our own results. In the remainder of this section, we describe our implementation of these three schedulers.

1) Global-SRPT: Shortest Remaining Processing Time is a traditional scheduling algorithm that provides optimal scheduling in terms of makespan in the single-server-single-queue scenario [24]. Global-SRPT is a variant of that mechanism for use in a multiple-servers-multiple-queues scenario. This is not a proposed scheduler from literature, but instead a simple adaptation of the existing algorithm for use in a new scenario, and as such we do not expect it to perform well. Its main purpose is to serve as a baseline for comparison of the other two algorithms.

For our implementation the scheduler keeps track of all jobs with unscheduled tasks and the total sum of the expected duration of all tasks of each job. Then if there is space available at any data center it schedules the longest task of the job with the shortest total processing time to a data center that is available and has the lowest time to download any required data. This process continues until there are no more data centers available or tasks to schedule.

2) SWAG: Workload Aware Geo-Distributed Scheduling (SWAG) was one of the first algorithms proposed for geo-

distributed scheduling. It was created to address the shortcomings of SRPT when applied to that same scenario, and uses a similar idea. One of SWAG's main distinguishing feature is that it restricts task placement to data centers that already have the required data. With that restriction in place, SWAG operates like Global-SRPT, but instead of selecting the job with the shortest total processing time, it selects the job with the shortest makespan, as if it were the only job being scheduled. The makespan is calculated as follows, with $W_{t,c}$ as the wait time to execute a task t on data center c, $D_{t,c}$ as the duration of task t on data center c, and $X_{t,c}$ as the time to transfer required data for task t to data center c: $\max_{t \in Job} \left(\min_{c \in C, X_{t,c}=0} W_{t,c} + D_{t,c} \right)$

3) GeoDis: GeoDis was proposed as an improvement over SWAG for job scheduling in geo-distributed data centers. GeoDis takes advantage of data transfers between data centers by accounting for data when calculating the expected completion time of a job, thus removing the restriction on task placement. Its implementation is similar to that of SWAG, but the removal of the locality restriction requires a modification of the expected completion time of each. Now the expected completion time considers not only current availability of all data centers, but also the duration of data migration to a data center that originally does not have the required data to execute a task. This results in the following formula for the expected completion time: $\max_{t \in Job} \left(\min_{c \in C} W_{t,c} + D_{t,c} + X_{t,c} \right)$

B. Experiments

Our experiments were designed to reproduce existing results and thus validate GDSim's operation. We compared the performance of Global-SRPT, SWAG, and GeoDis to verify the relations presented by Hung, Golubchik and Yu [23] and by Convolbo, Chou, Hsu and Chung [26]. We also compared our method for generation of synthetic data mixing features of different jobs against only sampling from existing jobs as proposed by Chen, Ganapathi, Griffith and Katz [15]. Each combination of parameters that we wanted to evaluate was executed with twenty four different slices of the three traces we used to verify consistency of the observed results. This amount was limited mainly by the size of the Facebook trace, which was the smallest of the three.

For the comparison experiment, we scheduled jobs using each of the three schedulers on each of the two topologies we described, using data from either of the three data sets we presented earlier. The purpose of this experiment was to verify that the performance improvements presented in their corresponding papers are still reproducible with our simulator. Namely, SWAG should perform better than using Global-SRPT, and GeoDis should perform better than SWAG when data co-location is factored in. We also repeated this experiment using the synthetic trace that we created in Section III-A to observe if it behaves as expected.

The purpose of our second experiment is to observe how our method of generating new traces compares to that of generating traces based on sampling existing jobs only. We schedule jobs from the Facebook trace by randomly sampling jobs and their characteristics independently and cross-generating the output accordingly. For all cases, we use the three aforementioned schedulers demonstrated in the smaller topology.

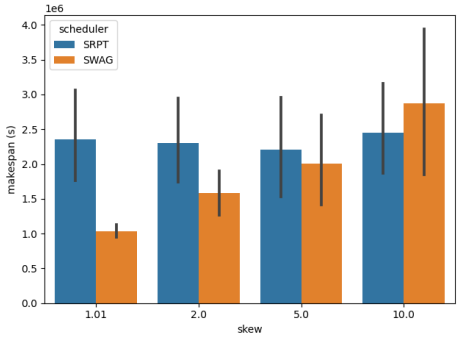

Fig. 4. Makespan with 95% confidence interval for SWAG and Global-SRPT for different file distribution skews in the larger topology using the Facebook trace.

V. Results and Discussion

In this section we will first look at the results from the experiments reproducing the reported performance for SWAG and GeoDis. We will follow with an examination of the results found when using synthetic data created according to the method we described in Section III-A.

A. Results Using Real Traces

We compare SWAG and Global-SRPT using the Facebook traces. Our results in Fig. 4 show SWAG performing better than Global-SRPT for most configurations as reported in SWAG's original paper [23]. However, when data files are more concentrated in fewer locations, SWAG's makespan deteriorates and, in some cases, is outperformed by Global-SRPT, a trend not revealed previously [23].

When comparing GeoDis and SWAG, we expect that GeoDis would yield superior performance thanks to data migration [26]. Fig. 5 shows results from executions in the larger topology using the Facebook trace. We observe that, as expected, SWAG's latency deteriorates as data files have fewer replicas, but under certain conditions, SWAG's makespan was similar to that of GeoDis. Reflecting on how SWAG compares to GeoDis, we observe that GeoDis performs better with regards to job latency when there is high concentration of per-job data within the same location, which is consistent with what was reported in the GeoDis paper [26]. When data files are spread out across more data centers, however, both schedulers exhibit similar behavior, i.e., they yield similar job latencies, which has not been previously reported [26].

B. Results Using Synthetic Traces

Using synthetic traces generated according to Section III-A, we observe again that GeoDis outperforms SWAG and Global-SRPT with regards to makespan when there is high resource utilization and less file replication, as shown in Fig. 6, with

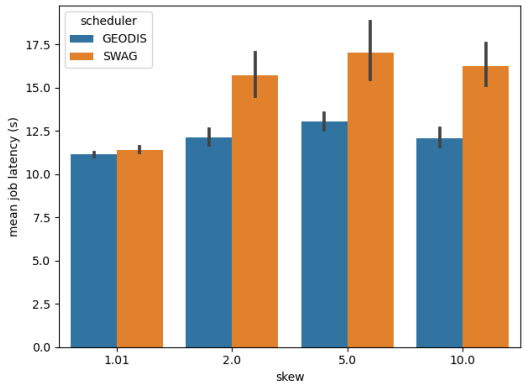

Fig. 5. Mean latency with 95% confidence interval for SWAG and GeoDis across different file distributions skews in the larger topology using the Facebook trace.

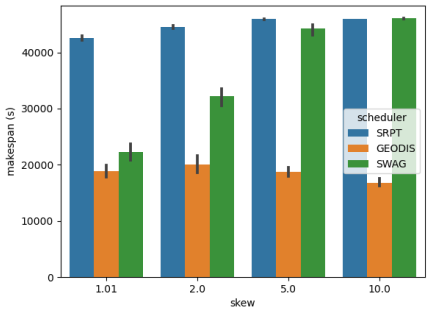

Fig. 6. Makespan with 95% confidence interval for Global-SRPT, SWAG and GeoDis across different file distribution skews in the smaller topology using the synthetic trace.

similar results when considering job latency as shown in Fig. 7.

We also run experiments to show the difference between job sampling and attribute sampling when generating synthetic traces. Fig. 8 compares GeoDis' makespan performance using the original Facebook trace against synthetic versions of the same trace generated via job sampling and attribute sampling. Our results show that attribute sampling yields higher deviation from the results obtained when using the original trace. This is an expected consequence of using attribute sampling since it aims at generating a wider variety of workloads yet preserving realistic features.

In summary, our experiments allowed us to validate GDSim by reproducing results reported in [23] and [26]. Additionally, we also show that, by being able to subject schedulers to a wide range of scenarios and workloads, GDSim can reveal behavior and performance trends not previously observed.

VI. Conclusion

This paper introduced GDSim, an open-source job scheduling simulation platform for geo-distributed data centers. GDSim's main goal is to provide a level playing field environment for evaluating different geo-distributed job schedulers to

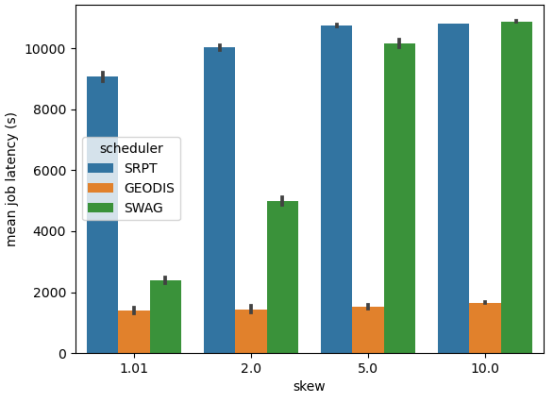

Fig. 7. Mean job latency with 95% confidence interval for Global-SRPT, SWAG and GeoDis across different file distribution skews in the smaller topology using the synthetic trace.

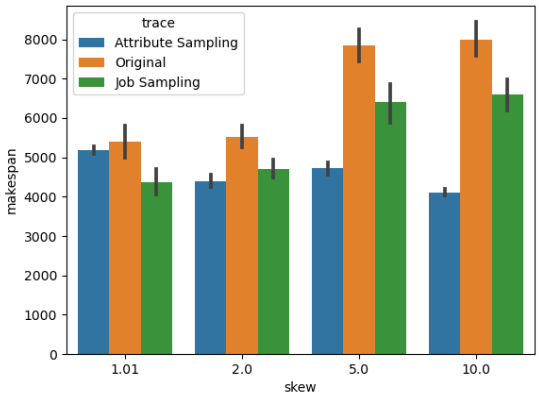

Fig. 8. Comparison of GeoDis performance in terms of makespan using the Facebook trace and two synthetic traces.

facilitate benchmarking, comparative evaluation efforts as well as development of new schedulers. Like any simulation platform, GDSim is meant to complement testbed experimentation as a tool to fast track evaluation in controlled and reproducible environments and can be instrumental to help guide the design of new geo-distributed job scheduling policies.

To showcase and validate GDSim, we used it to reproduce results from two existing scheduler proposals, namely SWAG [23] and GeoDis [26]. We also proposed a mechanism for generating synthetic traces from existing ones with the ability of mixing different workload characteristics, which allows exposing schedulers to a richer set of conditions and thus can uncover previously unknown behavior and trends.

As future work, we plan to implement other schedulers, further develop the job model to allow more distinct scenarios, and integrate GDSim with a network simulator in order to more realistically account for network dynamics.

REFERENCES

[1] G. D. Centers, "Discover our data center locations," 2020. Posted at https://www.google.com/about/datacenters/locations/.

[2] M. Azure, "Azure geographies," 2020. Posted at https://azure.microsoft.com/en-us/global-infrastructure/geographies/.

[3] F. Engineering, "Data centers year in review," 2018. Posted at https://engineering.fb.com/data-center-engineering/data-centers-2018/.

[4] N. Bansal and M. Harchol-Balter, "Analysis of srpt scheduling: Investigating unfairness," *SIGMETRICS Perform. Eval. Rev.*, vol. 29, p. 279–290, June 2001.

[5] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, p. 1–7, Aug. 2011.

[6] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, " Multi2Sim: A Simulation Framework for CPU-GPU Computing ," in *Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2012.

[7] S. Vincent, J. Montavont, and N. Montavont, "Implementation of an ipv6 stack for ns-3," in *VALUETOOLS*, 2008.

[8] P. Kugler, P. Nordhus, and B. Eskofier, "Shimmer, cooja and contiki: A new toolset for the simulation of on-node signal processing algorithms," in *2013 IEEE International Conference on Body Sensor Networks*, pp. 1–6, 2013.

[9] D. Klusáček, v. Tóth, and G. Podolníková, "Complex job scheduling simulations with alea 4," in *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques*, SIMUTOOLS'16, (Brussels, BEL), p. 124–129, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[10] D. Klusácek and H. Rudová, "Alea 2: job scheduling simulator," in *SimuTools*, 2010.

[11] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, (New York, NY, USA), p. 351–364, Association for Computing Machinery, 2013.

[12] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, (New York, NY, USA), Association for Computing Machinery, 2012.

[13] G. Sharma and A. Ganpati, "Performance evaluation of fair and capacity scheduling in hadoop yarn," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 904–906, 2015.

[14] A. K. Daniel Alves, Katia Obraczka, "GDsim," 2021. Posted at https://github.com/gdsim/gdsim.

[15] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The case for evaluating mapreduce performance using workload suites," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 390–399, 2011.

[16] K. Wang, Q. Zhou, S. Guo, and J. Luo, "Cluster frameworks for efficient scheduling and resource allocation in data center networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3560–3580, 2018.

[17] J. Kołodziej and S. U. Khan, "Data scheduling in data grids and data centers: A short taxonomy of problems and intelligent resolution techniques," in *Transactions on Computational Collective Intelligence X* (N.-T. Nguyen, J. Kołodziej, T. Burczyński, and M. Biba, eds.), pp. 103–119, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[18] T. Adhikary, A. K. Das, M. A. Razzaque, and A. M. J. Sarkar, "Energy-efficient scheduling algorithms for data center resources in cloud computing," in *2013 IEEE 10th International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pp. 1715–1720, 2013.

[19] F. Giroire, N. Huin, A. Tomassilli, and S. Pérennes, "When network matters: Data center scheduling with network tasks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 2278–2286, 2019.

[20] M. Halappanavar, M. Schram, L. de la Torre, K. Barker, N. R. Tallent, and D. J. Kerbyson, "Towards efficient scheduling of data intensive high energy physics workflows," in *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science*, WORKS '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[21] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, (Boston, MA), pp. 629–647, USENIX Association, Mar. 2017.

[22] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," *SIGCOMM Comput. Commun. Rev.*, vol. 45, p. 421–434, Aug. 2015.

[23] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, (New York, NY, USA), p. 111–124, Association for Computing Machinery, 2015.

[24] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, no. 3, pp. 687–690, 1968.

[25] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, 2016.

[26] M. W. Convolbo, J. Chou, C.-H. Hsu, and Y. C. Chung, "Geodis: Towards the optimization of data locality-aware job scheduling in geo-distributed data centers," *Computing*, vol. 100, p. 21–46, Jan. 2018.

[27] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, (New York, NY, USA), p. 351–364, Association for Computing Machinery, 2013.

[28] A. Hadoop, "Yarn scheduler load simulator," 2014. Posted at https://hadoop.apache.org/docs/r1.4.1/hadoop-sls/SchedulerLoadSimulator.html.

[29] Amazon, "Amazon Web Services," 2006. Posted at https://aws.amazon.com/.

[30] J. Wilkes, "More Google cluster data." Google research blog, Nov. 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[31] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[32] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *Proceedings of the International Symposium on Quality of Service*, IWQoS '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[33] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, p. 3–14, Aug. 2013.

[34] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Eighth workshop on computer architecture evaluation using commercial workloads*, pp. 1–8, 2005.

[35] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, and M. Wolf, "Monalytics: Online monitoring and analytics for managing large scale data centers," in *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, (New York, NY, USA), p. 141–150, Association for Computing Machinery, 2010.

[36] V. K. Naik, K. Beaty, N. Vogl, and J. Sanchez, "Workload monitoring in hybrid clouds," in *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 816–822, 2013.

[37] V. Jalaparti, P. Bodik, I. Menache, S. Rao, K. Makarychev, and M. Caesar, "Network-aware scheduling for data-parallel jobs: Plan when you can," *SIGCOMM Comput. Commun. Rev.*, vol. 45, p. 407–420, Aug. 2015.