2022 IEEE 40th International Conference on Computer Design (ICCD) | 978-1-6654-6186-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCD56317.2022.00103

Tear Up the Bubble Boom: Lessons Learned From a Deep Learning Research and Development Cluster

Zehua Yang*†, Zhisheng Ye*†, Tianhao Fu*, Jing Luo¶, Xiong Wei¶,
Yingwei Luo*†, Xiaolin Wang*†, Zhenlin Wang‡, Tianwei Zhang§
*Peking University †Peng Cheng Laboratory ‡Michigan Tech University
§Nanyang Technological University ¶Wuhan Textile University
yzh182_@stu.pku.edu.cn, yezhisheng@pku.edu.cn, tianhaofu@stu.pku.edu.cn, luojing0125@163.com, wx_wh@wtu.edu.cn
{lyw,wxl}@pku.edu.cn, zlwang@mtu.edu, tianwei.zhang@ntu.edu.sg

***Abstract*—With the proliferation of deep learning, there exists a strong need to efficiently operate GPU clusters for deep learning production in giant AI companies, as well as for research and development (R&D) in small-sized research institutes and universities. Existing works have performed thorough trace analysis on large-scale production-level clusters in giant companies, which discloses the characteristics of deep learning production jobs and motivates the design of scheduling frameworks. However, R&D clusters significantly differ from production-level clusters in both job properties and user behaviors, calling for a different scheduling mechanism. In this paper, we present a detailed workload characterization of an R&D cluster, `CloudBrain-I`, in a research institute, Peng Cheng Laboratory. After analyzing the fine-grained resource utilization, we discover a severe problem for R&D clusters, resource underutilization, which is especially important in R&D clusters while not characterised by existing works. We further investigate two specific underutilization phenomena and conclude several implications and lessons on R&D cluster scheduling. The traces will be open-sourced to motivate further studies in the community.**

***Index Terms*—Deep Learning , GPU cluster, Trace Analysis**

I. Introduction

Recent years have witnessed the prosperity of deep learning (DL) development and applications in every aspect of our daily life, including image classification, recommendation systems, text generation, etc. Such advancement also motivates the development of infrastructures for DL production and research in giant IT companies, research institutes and universities. It is common for these organizations to build up and operate shared GPU clusters to serve the DL workloads. In these clusters, one indispensable component is the scheduler, which plays a significant role on guaranteeing the job performance [1], [2], [3], resource utilization [4], [5], and user experience [6], [7], [8], [9].

Different from production-level GPU clusters, R&D GPU clusters in research and education institutes exhibit distinct cluster and job characteristics, bringing unique challenges for workload scheduling. We investigate an representative `CloudBrain-I` from the Peng Cheng Laboratory, which supports over 500 student researchers and staffs for AI research. We compare it with production-level clusters to summarize the following differences.

1. Computing resources. Large-scale production-level GPU clusters are usually equipped with customized high-bandwidth inter-node networking architecture [10], [11] and broad coverage of diverse GPU types. In contrast, small-scale R&D clusters usually provide relatively lower interconnection bandwidth via commodity InfiniBand or Ethernet and relatively limited generations of GPUs.

2) Mixed and diverse job properties. R&D clusters usually have a mix of jobs with different properties, including different types of workloads (e.g., preprocessing, training, inference) and execution environments. In contrast, production jobs in commercial IT companies are more homogeneous within a user, isolated from other users' jobs in terms of resource allocation, and vary significantly across different users. Due to the emerging need for debugging and the feedback-driven characteristic of DL jobs[1], users are enthusiastic about *interactive debugging* in DL R&D, resulting in remarkable proportion of these jobs. State-of-the-art GPU schedulers in production-level GPU clusters are already co-designed with a customized uniform DL framework, which are not practical for R&D clusters. For example, giant companies can afford to design DL frameworks or communication frameworks from scratch and integrate them with cluster scheduling, e.g., Pollux [3], BytePS [12], [13], and Bagua [14] for training. R&D cluster administrators often fall behind in supporting users with a wide variety of frequently-updated DL frameworks and software versions, such as requiring PyTorch version later than 1.9 for elastic training, specific GPU driver versions to support new GPUs, etc.

Except for a limited number of studies considering R&D clusters [15], [16], most research on DL scheduling focuses on production-level clusters. Unfortunately, direct application of these giant companies' experiences to the R&D clusters does not bring promising rewards due to the ignorance of R&D job characteristics. Existing schedulers from production-level GPU clusters usually model DL workloads as long-term offline processes, and the corresponding strategies may not be applicable for interactive debugging jobs in R&D clusters. We perform an in-depth analysis about the fine-grained resource utilization of `CloudBrain-I`, and find that even though the cluster-wide resource occupancy seems high, the actual job-level resource utilization is severely low. Although resource underutilization is also common in production-level clusters, we discover two unique causes leading to this issue in R&D

2576-6996/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCD56317.2022.00103

clusters due to the characteristic of R&D jobs. They are reflected in both spatial and temporal aspects. The spatial aspect refers to that many jobs could not achieve a high utilization on the allocated resource (Section IV-B), while the temporal aspect indicates that there exist idle time slots during the job lifetime (Section IV-C). Although several public traces of production-level GPU clusters are available for analysis and evaluation, the lack of such information about R&D jobs, especially on fine-grained resource utilization, hinders the scheduler design for R&D clusters.

To comprehensively analyze the job characteristics in R&D clusters, we collect a 328-day trace consisting of job and cluster information in `CloudBrain-I`. Unlike existing public traces in production-level clusters, we record not only the job lifecycle, resource requirements, and resource occupancy in the cluster but also detailed fine-grained resource usage of jobs including CPU, memory, and GPU. Based on the analysis of these fine-grained resource usage, we reveal the usage patterns and underutilization problems of GPU resources for R&D jobs. Regarding these phenomena and problems, we present new implications and lessons for cluster scheduler design. We believe that these discoveries and conclusions are general to other R&D clusters.

In summary, this paper makes the following contributions:

- We perform a thorough trace collection and analysis from both job and cluster levels in `CloudBrain-I`, a DL R&D GPU cluster, which is not well mentioned and studied in prior works. The trace from `CloudBrain-I` will be publicly available soon to benefit the community of DL system design and evaluation[1].
- We analyze an undercover but severe problem, job-level resource underutilization, from spatial and temporal aspects in R&D clusters, and identify severe causes.
- We present implications and lessons learned for effective R&D cluster scheduling .

II. Background

In this section, we introduce the basic information about the compute nodes and jobs in the `CloudBrain-I` cluster. Then we describe our methodology of trace collection, followed by the comparison with existing public traces of DL clusters.

A. Architecture of `CloudBrain-I`

`CloudBrain-I` is a GPU cluster dedicated for DL research and development in a research institute, Peng Cheng Laboratory. This cluster consists of 16 CPU nodes and 110 GPU nodes, with a total number of 1100 GPUs[2]. As shown in Table I, the cluster has 18 DGX-1s and 30 DGX-2s nodes, with each one containing 8 and 16 V100 GPUs. All the V100 GPUs have 32GB memory and slightly different clock speeds, memory frequency and power consumption. Additionally, the cluster has some customized nodes with RTX 2080, RTX 2080Ti and T4 GPUs, mainly for the debugging purpose. Different from the cluster trace analysis in previous works (e.g., Alibaba [5], SenseTime [17], Microsoft [18]) which mainly focused on the giant IT companies, the size of `CloudBrain-I` is smaller, with very limited types of heterogeneous NVIDIA GPUs. There are also 16 CPU machines in `CloudBrain-I`. Since we do not focus on CPU tasks, they are not described in detail here. There are 520 users in `CloudBrain-I`, most of which are students and researchers from different universities. A user account may be shared by multiple persons in the same research group, submitting jobs with different characteristics of resource usage. Besides, students are a fast-changing group along with the graduation of seniors and entrance of freshmen. Newcomers may be unfamiliar with the cluster, and their job submissions can lead to anomalies in the resource utilization, which will be discussed in detail in Section IV.

TABLE I
CONFIGURATIONS OF `CloudBrain-I`.

GPU Type	# of GPUs	# of nodes	# of CPUs	Mem (GiB)
V100-SXM2	8	24	96	1536
V100-SXM2-LS	8	18	80	512
V100-SXM3	16	30	96	1536
T4	8	26	40/80	384
RTX 2080	4	7	40/80	384
RTX 2080Ti	8	5	80	128
None	0	16	80/96	384/512/768

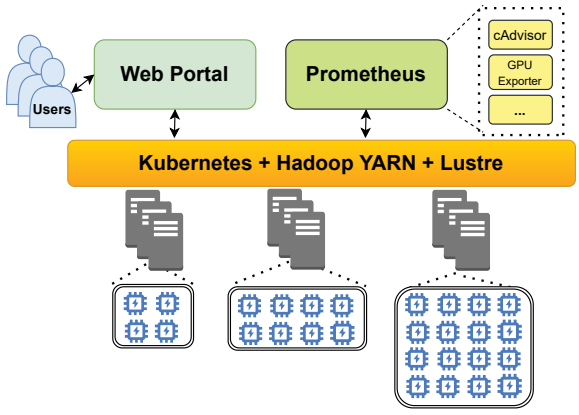


Fig. 1. `CloudBrain-I` Architecture

Figure 1 illustrates the key cluster architecture of `CloudBrain-I`, including the scheduler, monitoring system, and storage system. `CloudBrain-I` deploys the open-sourced scheduling framework OpenI-Octopus [19] on top of Kubernetes [20] and a set of exporters on every node. These exporters gather key metrics related to resource usage and system events like GPU utilization on the node at the frequency of seconds. Such information is then collected and stored by Prometheus [21], a time series database (TSDB), deployed along with the cluster. All the information about the resource utilization in the trace is dumped from Prometheus, which will be detailed in the rest of this paper. `CloudBrain-I` also leverages the parallel file system Lustre to satisfy the

[1]Please refer to the dataset: https://git.openi.org.cn/potato/ICCD-data

[2]The statistics were collected on January 19th, 2022. A few nodes in the cluster were drained for maintenance and temporarily unavailable during the trace collection period. This does not affect the conclusions in our analysis.

I/O requirements of DL jobs during the concurrent execution. Datasets which are public or manually uploaded by users in advance are stored on Lustre and available during the job execution. Despite the performance of inter-node network communication is crucial for the distributed training, for R&D clusters like `CloudBrain-I`, only part of nodes are equipped with EDR InfiniBand. Users need to explicitly request InfiniBand during the job submission. Otherwise, they are more likely to be assigned to a node with Ethernet only.

B. Workloads in `CloudBrain-I`

Users submit DL jobs to `CloudBrain-I` through the web portal, which also provides a graphical interface to view the configuration and current status of their jobs. User can also terminate their jobs manually at any time. During the submission, users should specify the execution environment, launch command, the number of tasks, the requested amount of GPUs and CPUs per task, and host memory. `CloudBrain-I` automatically schedules the jobs and runs all the tasks of each selected job simultaneously (gang scheduling).

The workloads in `CloudBrain-I` cover various types of jobs at every stage of the R&D pipeline, showing the mixed and diverse characteristics. Some CPU jobs are submitted for preprocessing and may incur more I/O-intensive operations such as dataset loading and decompression, environment installation, etc. GPU jobs are mainly for DL training and inference. DL training is an iterative process where a fixed size of input samples (mini-batch) are fed into the model. The output is computed after forward propagation, and the gradient is obtained by backward propagation based on the difference (loss) between the output and actual result. The gradient is then used for updating the model parameters. Both forward and backward propagation heavily relies on GPUs for parallel computation. For multi-GPU jobs, the gradient needs to be communicated among all the workers, which is communication-intensive. Different from the production-level inference systems in IT companies which focus on balancing the model accuracy with the latency constraints, DL R&D inference jobs mainly evaluate models on a relatively fixed validation set and thus may incur more stable execution in a repeated manner.

The models and frameworks used by these DL R&D jobs also exhibit diversity and complexity. Inferred from the names of jobs and their docker images in the trace, the jobs consist of many kinds of neural network models, such as convolutional neural network (e.g., ResNet [22], VGG [23]), recurrent neural networks (e.g., LSTM [24]), and transformer-based models (e.g., BERT [25]). They are implemented by TensorFlow [26], PyTorch [27], and some other popular DL frameworks. Such complexity and diversity exhibited in DL R&D jobs and the need for interactive debugging bring challenges to the job scheduling.

C. Trace Collection and Information

The per-job information in the collected trace, including when the job is submitted, started, and finished, along with the fine-grained resource utilization. The resource-related information is collected and stored in Prometheus mentioned in Section II-A, from which we obtain the time series of resource utilization of CPUs, host memory, GPU SM, and GPU memory at an interval of 15 seconds during the job's entire lifetime. The metric of GPU utilization is provided by NVIDIA GPU Exporter [28]. GPU utilization is collected separately for each card in multi-GPU jobs. The resource utilization information in the collected job trace reveals the underutilization issue, which will be detailed in Section IV.

D. Comparison with Existing Public Traces

Over the past few years, several studies have analyzed the public traces and characterized the DL jobs from the scheduler's perspective. One important public trace is Philly [18] from a Microsoft cluster, which supports the production DL workloads in 2017 and has been deprecated now. The characteristics of DL workloads have changed dramatically since then. Other public traces in recent years include Helios [17] and PAI [5], which have inspired the designs of many DL schedulers [8], [9]. However, all these traces are from production-level GPU clusters, which are significantly different from DL R&D clusters. A recent work [29] featuring scheduling optimization for R&D clusters performs job analysis without releasing the trace to the public. To the best of our knowledge, our work provides the first publicly available DL R&D cluster trace that covers the analysis of fine-grained resource usage.

Table II compares our trace with existing public traces. These traces serve as important knowledge for workload characterization and system design for DL clusters. Specifically, the size of `CloudBrain-I` is smaller with approximately 1,100 GPUs, while Helios and PAI have more than 6,000 GPUs, and Philly has more than 2,400 GPUs in 2017. Due to the smaller cluster size, the largest GPU job in `CloudBrain-I` requests 768 GPUs, much smaller than the 2,048-GPU job in Helios.

The composition and properties of jobs in `CloudBrain-I` also present different characteristics from those production-level GPU clusters. The users in `CloudBrain-I` are mainly interns or junior researchers in research institutes and universities, and thus have strong needs for debugging and experimental exploration. The submitted jobs range from all stages in the DL pipeline, thus being more diverse and error-prone. In contrast, production-level clusters are filled with mature and automated jobs that are primarily for model production.

Supporting interactive debugging jobs in `CloudBrain-I` also leads to longer job duration in the job trace due to the processing approach of these jobs. While users in other clusters submit interactive debugging jobs by dividing them into multiple consecutive short jobs (e.g., job attempts in Philly), the way users submit debugging jobs in `CloudBrain-I` results in significantly longer average job completion time.

TABLE II
COMPARISONS BETWEEN CLOUDBRAIN-I AND OTHER PRODUCTION-LEVEL TRACES.

	Year	Trace Duration	# of GPUs	# of GPU Jobs	Average Jobs Duration (s)	Scheduler	Institution
Philly [18]	2017	3 months	2490	103K	28329	YARN	Microsoft
Helios [17]	2020	6 months	6416	1.58M	13040	Slurm	SenseTime
PAI [5]	2020	2 months	6742	1.2M	4821	Fuxi	Alibaba
CloudBrain-I	2020-2021	10 months	1116	155K	42672	OpenI-Octopus	Peng Cheng Laboratory

III. WORKLOAD CHARACTERIZATION

In this section, we present detailed characterization about the DL jobs in the CloudBrain-I trace. We analyze the statistics about their duration, requested and utilized resources. We then show the explicit daily and weekly trends of job submissions, which highly reflect users' behaviors. Based on such analysis, we are surprised to find that the underscored metric, cluster-wide GPU occupancy, is not well applied to DL R&D clusters to demonstrate their GPU usage efficiency. The cluster-wide GPU occupancy concentrates on the resource allocation process to jobs, while resource underutilization at the job level is severely ignored. Motivated by the remarkable gap between users' requested and utilized resources, we will present more detailed analysis of resource utilization in Section IV.

A. Job Duration

Figure 2(a) compares the Cumulative Distribution Function (CDF) of the job duration between CloudBrain-I and previous traces (PAI [5], Helios [17] and Philly [18]). Similar as those production-level jobs traces, the job duration in CloudBrain-I is also widely distributed and long-tailed. The average and medium duration of jobs in CloudBrain-I is 42672s and 6182s respectively, which is much longer than previous works. There is a sharp increase at around 28,800s (8 hours) due to the setting of the maximum duration for debugging jobs.

B. Job Resource Usages

Figures 2(b) - 2(d) show the job-level requests and usages of GPUs, CPUs, and host memory resources in CloudBrain-I, respectively. We observe that the resource requests of these jobs also exhibit wide distributions: the majority of jobs request the minority of resources. Specifically, the 80*th* percentile of job resource requests are 8 CPUs, 2 GPUs, and 128GB host memory, while the maximum resources requested by one job are 3,840 CPUs, 768 GPUs, and 39,552GB host memory respectively, consuming nearly 80% of GPUs in the cluster.

We retrieve the task-level resource usage as the mean of the time series of GPU, CPU, and host memory usages. The resource usage of a job is the average of the resource usage of all its tasks. As shown in Figures 2(b) - 2(d), it is obvious that the resource usages (GPU, CPU, and host memory) of all the jobs are much lower and more smooth than the requested amounts. This indicates severe resource wastes, and will be detailed in Section IV.

We analyze the relationship between the requested numbers of GPUs and the GPU time, which is defined as the number of GPUs multiplied by the job duration. Figure 3 compares the CDFs of jobs and GPU time in terms of GPU resource demands. It shows about 60% of the jobs request one GPU. However, they only consume about 20% of the GPU time. Multi-GPU jobs consume most of the GPU time, which is similar to other GPU clusters like Helios [17].

IV. RESOURCE UNDERUTILIZATION

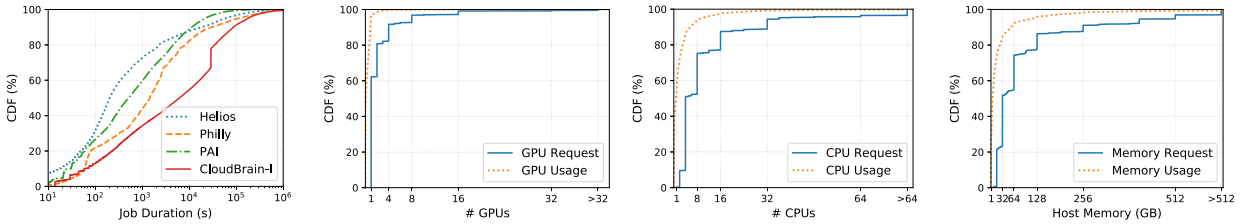
From Section III, we observe there exists a resource underutilization problem in CloudBrain-I. In this section, we make an in-depth analysis of this issue and elaborate the reason that leads to such underutilization. We summarize several typical anomalous resource usage patterns of jobs through quantitative analysis. These patterns together contribute to the underutilization issue. We also present some implications learned from this analysis. These lessons can also be applied to production-level clusters that suffer from the similar underutilization problem [5], [17].

A. Job-level Resource Underutilization

The GPU resources in CloudBrain-I are not fully utilized. Figure 4 shows the distributions of the average utilization of GPUs, GPU memory and CPUs among different jobs. We observe that jobs cannot utilize the allocated GPUs well. The average values of GPU utilization and GPU memory usage are **surprisingly low**, with approximately 70.57% of jobs consuming less than 50% of both GPU cards and GPU memory. To make matters worse, a significant percentage of jobs (54.57%) have very low average GPU utilization ($\leq$20% of both GPU cards and memory), clustered in the left bottom corner in Figure 4(a). The CPUs are also not fully utilized in CloudBrain-I and shows a mismatch with the GPU resource allocation. Figure 4(b) shows the CPU utilization distribution along with GPU utilization among jobs. There are 50.28% of jobs with the consumption of less than 20% of both GPU cards and CPUs, shown in Figure 4(b).

Implication #1: The utilization of GPUs, CPUs, and GPU memory shows heterogeneity among jobs, which increases the difficulty of job scheduling.

The dense areas in the top left and bottom right of Figure 4 suggest that a noticeable percentage of jobs have relatively **contrary** average utilization of GPUs versus CPUs, demonstrating the potential imbalance in resource allocation for these jobs. The CPU utilization of jobs with high GPU utilization



(a) Job Duration Distribution (b) GPU Request and Usage (c) CPU Request and Usage (d) Host Memory Request and Usage

Fig. 2. Job duration, resource requests and usages in `CloudBrain-I`.

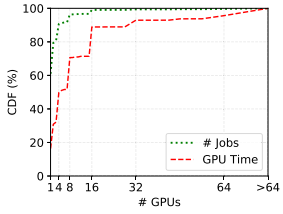
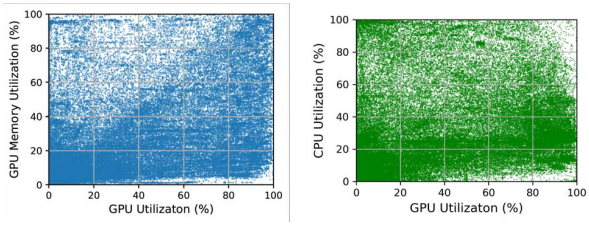
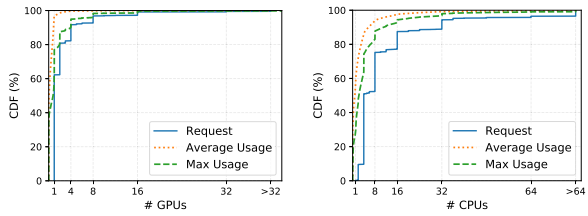


Fig. 3. Job GPU requests and GPU time distribution.

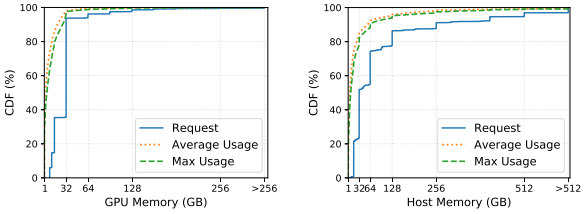


(a) GPU-GPU Memory (b) GPU-CPU

Fig. 4. Distributions of the average utilization of GPUs, CPUs, and GPU memory across different jobs. A point in the figure represents one job respectively. The x and y coordinates of the point represent the utilization of GPU cards and memory/CPUs in Figure (a)/(b).



(a) GPU (b) CPU



(c) GPU Memory (d) Host Memory

Fig. 5. Maximum and average resource usage distributions of GPUs, CPUs, GPU memory and host memory among jobs.

(≥80%) is mainly (67.91%) distributed from 10% to 40%. About 5% of jobs suffer from a severe mismatch, which have a utilization of greater than 80% for one resource and less than 20% for the other resource. This may be because that users are not familiar with the real resource requirements of their jobs, and thus request inappropriate configurations of CPU and GPU resources. The performance of DL workloads may also be affected due to their sensitivity to the resources [30].

Implication #2: The imbalanced utilization of various resources in `CloudBrain-I` is a common problem, wasting large amounts of expensive resources and threatening the job performance.

Figure 5 shows the resource usage and maximum utilization among jobs. Both the GPUs and CPUs are underutilized. The average utilization of GPUs, GPU memory, CPUs and host memory among jobs are 21.24%, 25.75%, 18.12%, 33.08% respectively. The median of maximum GPU and GPU memory usage are 0.78 and 2.4 GB respectively. As seen from this figure, the maximum demand for resources by jobs also tends to be lower than the requested resources. One reason for such mismatch between allocated and utilized resources lies on the approaches of resource allocation in the schedulers. Existing schedulers for DL training usually consider GPUs as the dominant resource in the cluster and focus on GPU allocation in achieving the desired scheduling objectives. The scheduler of `CloudBrain-I` follows this strategy and allocates other resources, including CPUs and main memory, proportionally based on the pre-defined resource configurations mentioned in Section II-B.

Implication #3: Low utilization and overestimation of resource requirements are common for jobs in `CloudBrain-I`. It exacerbates the waste of various computing resources and incentivizes the feasibility of GPU sharing.

B. Fluctuating Resource Utilization

The iterative characteristic of DL job execution leads to fluctuating usages of CPU and GPU resources, especially for GPUs, wasting resources **spatially**. During each mini-batch of DL training, besides the heavy parallel computation performed on the GPUs such as forward and backward propagation, a large amount of data are transferred for communication at the beginning and the end of the mini-batch, resulting in an iterative usage of GPUs in the training process. There are also many framework-level optimizations dedicated to synchronizing the computation and communication processes for more efficient GPU usage [31], [32]. However, the dynamic fluctuating usage of GPU resources brings challenges for DL workloads to saturate the GPU's compute capacity. Below we give detailed illustration of this resource utilization characteristic.

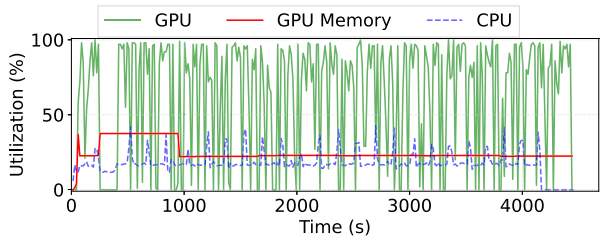


Fig. 6. The resource utilization of a task from the job trace. The x-axis denotes the time during the training. The blue, red and green lines represent the utilization of CPUs, GPU memory, and one allocated GPU for this task.

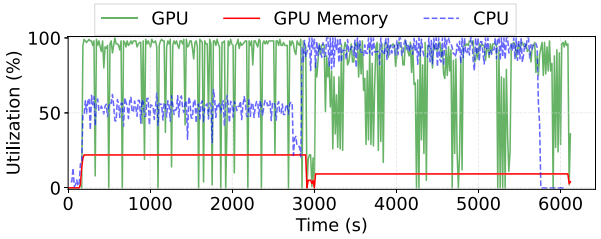
Figure 6 shows the utilization of GPUs, CPUs and GPU memory for an example task. It clearly shows the fluctuation of GPU utilization, caused by the DL framework behaviors. Specifically, DL frameworks (e.g., TensorFlow, PyTorch) need to reconcile the processes of data loading, transferring to GPUs, computation on GPUs, and copying back to CPUs for aggregation or synchronization. Therefore, it is hard for DL jobs to saturate the allocated computation resources (e.g., CPUs and GPUs) all the time. About 11.54% of the jobs in `CloudBrain-I` show a similar pattern of sharp and unstable fluctuations (utilization floats over 90% in most (over 75%) of intervals with 120s) in CPU or GPU utilization. This poses a pitfall that GPU sharing may lead to inter-job interference easily without considering the sharp fluctuations.

Implication #4: The fluctuating CPU and GPU utilization indicates the strong need for safe and efficient GPU sharing. Schedulers need to carefully consider fine-grained dynamic GPU sharing to maximize the utilization and minimize the interference across jobs.

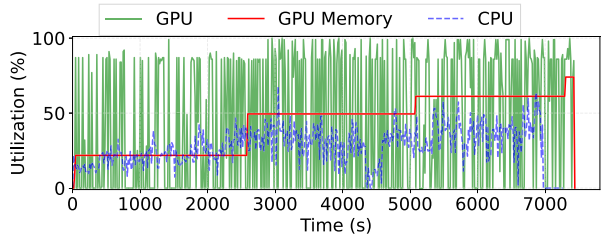
We define a stable phase as a period whose length is larger than a threshold t and the idle GPU memory is larger than another threshold m. Table III shows the ratio of stable phases in our traces under different thresholds (t, m). We consider the thresholds of (240s, 2G), and 87.02% GPU time satisfies this requirement. This shows that GPU memory is stable in most jobs' lifetime, which provides the opportunity to make a safe GPU sharing among jobs.

TABLE III
GPU TIME PERCENTAGE OF STABLE PHASES UNDER DIFFERENT LENGTH THRESHOLD AND IDLE GPU MEMORY THRESHOLD.

m (G) \ t (s)	60	120	240	480	960
1	92.28	91.77	90.86	89.65	88.21
2	88.41	87.91	87.02	85.83	84.45
4	82.90	82.43	81.59	80.48	79.22
8	70.68	70.27	69.51	68.52	67.42



(a) Job A with two stable phases: 200s – 2800s and 3000s – 6200s.



(b) Job B with three stable phases: 0s – 2600s, 2600s – 5100s, and 5100s – 7300s. The last period is too short ($\leq$240s) to be considered as a stable phase.

Fig. 7. Two jobs which have multiple stable phases.

A remarkable proportion (about 38.05%) of GPU jobs have multiple stable phases. Figure 7 shows two example jobs with multiple stable phases. Jobs may have different GPU memory requests in different phases during their lifetime. Over 20.97% of jobs have a memory gap greater than 1G and over 8.99% of jobs have a gap greater than 8G among different phases. Lots of GPU memory is wasted if the scheduler just allocates GPU memory according to the largest requirement. This poses a strong need for schedulers to detect the phases and allocate proper resources according to the changing demands when colocating jobs.

Implication #5: The stable phase provides feasibility for safe GPU sharing. The characteristic of jobs containing multiple stable phases requires the scheduler to dynamically adjust the GPU memory allocation.

C. Non-continuous GPU Utilization

In addition to the feature of resource underutilization, we perform further analysis of their resource usages from a **temporal** perspective. We find that many jobs cannot maintain an active GPU usage during their whole lifetime, but consume the expensive GPU resources in a non-continuous manner. To quantify this non-continuous usage phenomenon and analyze its severity and potential causes, we define the *idle slot* of a job as a period of length larger than a threshold, during which the job does not actually use any GPU resources (i.e., both GPU utilization and GPU memory usage are equal to 0). We set this threshold to 60s and calculate the percentage of idle slots contributing to the total GPU time.

Table IV describes the statistics about the idle slots of jobs in `CloudBrain-I`. We consider two types of jobs: debugging and non-debugging jobs. We also consider two GPU usage scenario: *idle-slot* means the job still uses GPUs although there exist some idle slots; *never-used* means the job never used part of the allocated GPUs during its lifetime. We compute the percentage of such GPU time among the GPU time of this job, as well as all the jobs in the cluster.

TABLE IV
THE STATISTIC RESULTS OF NON-CONTINUOUS GPU USAGE.

	% among	**Debugging**	**Non-debugging**	Total
Idle-slot GPU time	its own job	41.62	9.76	-
	all the jobs	1.77	9.35	11.12
Never-used GPU time	its own job	36.10	2.76	-
	all the jobs	1.54	2.64	4.18
Total	its own job	77.72	12.52	-
	all the jobs	3.31	11.99	15.30

We observe that the debugging jobs have a much higher ratio of idle slots than the non-debugging jobs due to their debugging properties. About 77.7% of the lifetime of these debugging jobs is wasted. They are likely the cause of the GPU machine time waste. However, the idle slots of non-debugging jobs contribute more than debugging jobs to the GPU machine time waste considering the total GPU time. For these non-debugging jobs, most of the GPU waste is caused by the idle slots among job duration.

1) Idle Slots among Job Lifetime: We look deeper into the idle slots of jobs in `CloudBrain-I`. They play the most important role in wasting GPU time for both debugging and non-debugging jobs. The idle slots of non-debugging jobs are possibly caused by the cold start overhead for initializing runtime environments and pre- or post-processing in the DL lifecycle. Since we do not have more runtime information (e.g., the time information corresponding to the model training stage) to reason about the idle slots, we only focus on the idle slots of debugging jobs.

Figure 8 shows the characteristics of the GPU utilization for one interactive debugging job over the time. During the first 3 hours, the user debugs and evaluates the job on the GPU interactively. It clearly shows that the GPU is not used between 0s to 600s, 1200s to 1800s, and 4400s to

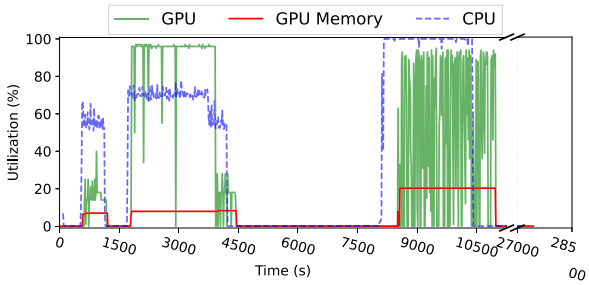


Fig. 8. Resource utilization of a debugging job. The period between 4500s to 9000s reflects its interactive property.

8500s. The job continues to run until it is terminated by the scheduler when the 8-hour quota is reached. Due to such interactive execution characteristics, all the debugging jobs in `CloudBrain-I` waste about 41.62% of the allocated GPU time. We also observe that nearly 80% of debugging jobs actually do not use GPUs among 80% of their lifetime. Existing scheduling algorithms ignore such characteristic and allocate exclusive GPUs to debugging jobs, which can waste GPU resources greatly. The long idle time of debugging jobs is also exacerbated by the fact that users forget to stop the debugging jobs. About 40.41% of the debugging jobs approach or exceed the maximum duration (8 hours).

Implication #6: The interactive and exploration nature of debugging jobs prevents them from enjoying high resource utilization, and causes great GPU resource waste. Schedulers for R&D clusters should be able to identify and apply different scheduling policies to such interactive jobs.

2) Never-used Allocated GPUs: Another phenomenon of GPU waste is the presence of completely unused GPUs in the job. This is not rare for both debugging and non-debugging jobs in `CloudBrain-I`. There are much more job submissions for testing the correctness of the program, than production development in R&D clusters. These testing jobs generally request certain amount of resources for execution. After deeper analysis of multi-GPU jobs, we find that there exists a remarkable proportion (about 24.01%) of DL jobs that only use part of the requested GPUs during their lifetime. Table V shows the abnormal number of jobs that waste at least one GPU entirely. One possible reason is that users could misconfigure the important parameters in distributed training frameworks and environments in R&D clusters. From a scheduling perspective, it is necessary to proactively monitor GPU usage of all the jobs and provide timely feedback to users.

Implication #7: The exploration nature of DL jobs in R&D cluster leads to frequent misusage of GPUs, thus wasting GPU resources. Schedulers for R&D clusters should be able to identify the abnormal resource usage scenarios and make dynamic GPU resource allocation for different jobs.

TABLE V
STATISTICS ABOUT THE JOBS WITH UNUSED GPUS.

# of GPUs requested	# of Jobs with unused GPUs	Ratio
1	24370	26.20%
2	6671	23.82%
4	2336	15.47%
8	1016	14.23%
16	553	17.43%
Other	1514	24.16%
Total	36460	24.01%

V. RELATED WORK

A. DL Trace Analysis of GPU Clusters

Recently released traces of DL training workloads are mainly from production-level clusters in giant IT companies. Philly [18] from Microsoft shows the influence of job locality on queuing delay and resource utilization, and identifies the percentage of different reasons for errors. Helios [17] from SenseTime shows the characteristics of cluster resource utilization and unfairness among users from the cluster, job, and user perspectives. Weng et al. [33] analyze the challenges in Alibaba's cluster PAI from temporal and spatial perspectives. Wang et al. [34] focus on the performance bottlenecks of jobs in Alibaba's PAI, which is not detailed in the paper because of the lack of public traces.

Different from those works, this paper fills in the missing part of DL workloads traces for R&D clusters. We collect and analyze a trace from `CloudBrain-I`, demonstrate different characteristics of R&D jobs, and provide guidance for scheduling of R&D clusters. Some findings and scheduling implications from prior works may also be applicable to R&D clusters due to some similarities between R&D jobs and production jobs.

B. Deep Learning Cluster Scheduling

Past years witness a wealth of research works to optimize the execution of DL training jobs in GPU clusters from different perspectives. To maximize the cluster efficiency, Gandiva [1] designs primitives for DL job packing and sharing, as well as introspective migration for the cluster scheduling. Antman [35] supports controlling fine-grained elastic usage of GPUs by grow-shrink, thus enabling sharing GPUs between jobs. Pollux [3] rearranges resources to different jobs to improve cluster-wide throughput with dynamic adjustment of batch sizes and learning rates. To minimize the average job completion time, Optimus [36] and Tiresias [2] predict job remaining time by learning the training progress till convergence and analytic modeling respectively. To satisfy the QoS requirements, Allox [37], $Gandiva_{fair}$ [6] and Gavel [38] target on heterogeneous clusters with different generations of GPUs or computation resources, while HiveD [4] considers the inter-GPU affinity as a type of resource for isolating and allocating GPUs. Themis [7] and ASTRAEA [9] measure the impact of placement policies on the job performance for fairness enhancement. Chronus [8] adopts the intra-job predictability to guarantee the deadline of DL jobs.

However, these works focus on specific optimization for DL R&D clusters rather than general workload characterization. We provide a thorough analysis and characterization for the workloads, which could inspire other works to optimize the management of DL jobs in R&D clusters from diverse perspectives.

VI. CONCLUSION

In this paper, we collect and analyze job traces from a deep learning research and development cluster (`CloudBrain-I`) over 10 months. We uncover the underutilization of different resources (GPUs, CPUs, host memory), the most severe problem for R&D jobs. We also analyze the causes of the underutilization, including the fluctuating GPU usages of DL jobs, the interactive execution of debugging jobs, abnormal resource consumption due to users, and the mismatching between resource allocation and utilization. We conclude the implications and lessons which could inspire new solutions to this problem. The trace will be publicly available for further investigation and benefit the deep learning scheduling community.

ACKNOWLEDGEMENTS

The research is supported in part by the National Science Foundation of China (Nos. 62032001, 62032008) and PKU-Baidu Fund 2020BD001. This study is also supported under the RIE2020 Industry Alignment Fund–Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contributions from the industry partner(s).

REFERENCES

[1] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," in *OSDI*, 2018.

[2] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. Liu, and C. Guo, "Tiresias: A gpu cluster manager for distributed deep learning," in *NSDI*, 2019.

[3] A. Qiao, S. K. Choe, S. J. Subramanya, W. Neiswanger, Q. Ho, H. Zhang, G. R. Ganger, and E. P. Xing, "Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning," in *OSDI*, 2021.

[4] H. Zhao, Z. Han, Z. Yang, Q. Zhang, F. Yang, L. Zhou, M. Yang, F. C. Lau, Y. Wang, Y. Xiong, and B. Wang, "Hived: Sharing a GPU cluster for deep learning with guarantees," in *OSDI*, 2020.

[5] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and D. Yu, "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *NSDI*, 2022.

[6] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing efficiency and fairness in heterogeneous gpu clusters for deep learning," in *EuroSys*, 2020.

[7] K. Mahajan, A. Balasubramanian, A. Singhvi, S. Venkataraman, A. Akella, A. Phanishayee, and S. Chawla, "Themis: Fair and efficient gpu cluster scheduling," in *NSDI*, 2020.

[8] W. Gao, Z. Ye, P. Sun, Y. Wen, and T. Zhang, "Chronus: A novel deadline-aware scheduler for deep learning training jobs," in *SoCC*, 2021.

[9] Z. Ye, P. Sun, W. Gao, T. Zhang, X. Wang, S. Yan, and Y. Luo, "Astraea: A fair deep learning scheduler for multi-tenant gpu clusters," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2021.

[10] J. Dong, Z. Cao, T. Zhang, J. Ye, S. Wang, F. Feng, L. Zhao, X. Liu, L. Song, L. Peng, Y. Guo, X. Jiang, L. Tang, Y. Du, Y. Zhang, P. Pan, and Y. Xie, "Eflops: Algorithm and system co-design for a high performance distributed training platform," in *HPCA*, 2020.

[11] N. Corporation, "Nvidia dgx superpod," 2020. [Online]. Available: https://www.nvidia.com/en-us/data-center/dgx-superpod

[12] Y. Peng, Y. Zhu, Y. Chen, Y. Bao, B. Yi, C. Lan, C. Wu, and C. Guo, "A generic communication scheduler for distributed dnn training acceleration," in *SOSP*, 2019.
[13] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *OSDI*, 2020.
[14] S. Gan, X. Lian, R. Wang, J. Chang, C. Liu, H. Shi, S. Zhang, X. Li, T. Sun, J. Jiang, B. Yuan, S. Yang, J. Liu, and C. Zhang, "Bagua: Scaling up distributed learning with system relaxations," 2021.
[15] Z. Chen, W. Quan, M. Wen, J. Fang, J. Yu, C. Zhang, and L. Luo, "Deep learning research and development platform: Characterizing and scheduling with qos guarantees on gpu clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 34–50, 2020.
[16] Z. Chen, "Rifling: A reinforcement learning-based gpu scheduler for deep learning research and development platforms," *Software: Practice and Experience*, 2021.
[17] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in *SC*, 2021.
[18] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of large-scale multi-tenant gpu clusters for dnn training workloads," in *ATC*, 2019.
[19] open-intelligence contributors, 2020. [Online]. Available: https://github.com/open-intelligence/OpenI-Octopus
[20] Kubernetes contributors, 2021. [Online]. Available: https://kubernetes.io/
[21] Prometheus contributors, 2022. [Online]. Available: https://prometheus.io/
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735
[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.
[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016.
[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, 2019.
[28] NVIDIA GPU Prometheus Exporter contributors, 2022. [Online]. Available: https://github.com/mindprince/nvidia_gpu_prometheus_exporter
[29] S. Wang, P. Yang, Y. Zheng, X. Li, and G. Pekhimenko, "Horizontally fused training array: An effective hardware utilization squeezer for training novel deep learning models," in *MLSys*, 2021.
[30] J. Mohan, A. Phanishayee, A. Raniwala, and V. Chidambaram, "Analyzing and mitigating data stalls in DNN training," *Proc. VLDB Endow.*, vol. 14, no. 5, pp. 771–784, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p771-mohan.pdf
[31] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, 2019.
[32] P. Sun, Y. Wen, R. Han, W. Feng, and S. Yan, "Gradientflow: Optimizing network performance for large-scale distributed dnn training," *IEEE Transactions on Big Data*, 2019.
[33] M. Wang, C. Meng, G. Long, C. Wu, J. Yang, W. Lin, and Y. Jia, "Characterizing deep learning training workloads on alibaba-pai," in *IISWC*, 2019, pp. 189–202.
[34] W. Mengdi, M. Chen, L. Guoping, W. Chuan, Y. Jun, L. Wei, and J. Yangqing, "Characterizing deep learning training workloads on alibaba-pai," *CoRR*, vol. abs/1910.05930, 2019. [Online]. Available: http://arxiv.org/abs/1910.05930
[35] W. Xiao, S. Ren, Y. Li, Y. Zhang, P. Hou, Z. Li, Y. Feng, W. Lin, and Y. Jia, "Antman: Dynamic scaling on GPU clusters for deep learning," in *OSDI*, 2020.
[36] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: an efficient dynamic resource scheduler for deep learning clusters," in *EuroSys*, 2018, pp. 1–14.
[37] T. N. Le, X. Sun, M. Chowdhury, and Z. Liu, "Allox: Compute allocation in hybrid clusters," in *EuroSys*, 2020.
[38] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in *OSDI*, 2020.