Brief report on computer systems

Brief Introduction

This report is mainly about some information related to OS and parallel and distributed system. In the first part of this report, some top conferences are listed and I collects several papers that I am interested. In the second part, I read and understander 4 papers in detail. In addition, I make a summarization of each paper according to my understanding. In the final part, I make a conclusion about what I gained from this report as well as a reflection on what I am interested in and what I am good at.

The areas I am interested in are:

1. Distributed and Parralel computing
2. Operating Systems

- Developing a better OS/Parallel computing system for machine learning/deep learning
- Applying mechine learning techniques to improve performance of OS / Parallel and distributed system

---

The best conferences

1. Distributed and Parralel computing
    - PPoPP: Principles and Practice of Parallel Programming
      https://dblp.org/db/conf/ppopp/index.html
    - PACT: Intl Conf on Parallel Arch and Compil Tech
      https://dblp.org/db/conf/IEEEpact/index.html
    - IPDPS: IEEE Intl Parallel and Dist Processing Symp
      https://dblp.org/db/conf/ipps/index.html
    - ICPP: Intl Conf on Parallel Processing
      https://dblp.org/db/conf/icpp/index.html
    - Euro-Par: European Conf. on Parallel Computing
      https://dblp.org/db/conf/europar/index.html
2. Operating Systems
    - SOSP: ACM SIGOPS Symp on OS Principles
      https://dblp.org/db/conf/sosp/index.html
    - OSDI: Usenix Symp on OS Design and Implementation
      https://dblp.org/db/conf/osdi/index.html

Some studies in the above conferences

To know more about the current studies in systems and explore what I am actually interested in, I read some papers and summarize some contents of them.

- The first paper is about building a communication scheduler for distributed DNN trainning
    - Peng, Y., Zhu, Y., Chen, Y., Bao, Y., Yi, B., Lan, C., ... & Guo, C. (2019, October). A generic communication scheduler for distributed dnn training acceleration. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (pp. 16-29).
- The second paper is a discussion about about the possibilities of enhancing the performance of OS using meachine learning
    - Zhang, Y., & Huang, Y. (2019). " Learned" Operating Systems. ACM SIGOPS Operating Systems Review, 53(1), 40-45.
- The third paper is applying desicion tree to the process scheduling of Linux
    - Negi, A., & Kumar, P. K. (2005, November). Applying machine learning techniques to improve linux process scheduling. In TENCON 2005-2005 IEEE Region 10 Conference (pp. 1-6). IEEE.
- The fourth paper is about combining data parallelism and pipeline parallelism for DNN trainning.
    - Fan, S., Rong, Y., Meng, C., Cao, Z., Wang, S., Zheng, Z., ... & Lin, W. (2021, February). DAPPLE: A pipelined data parallel approach for training large models. In Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (pp. 431-445).

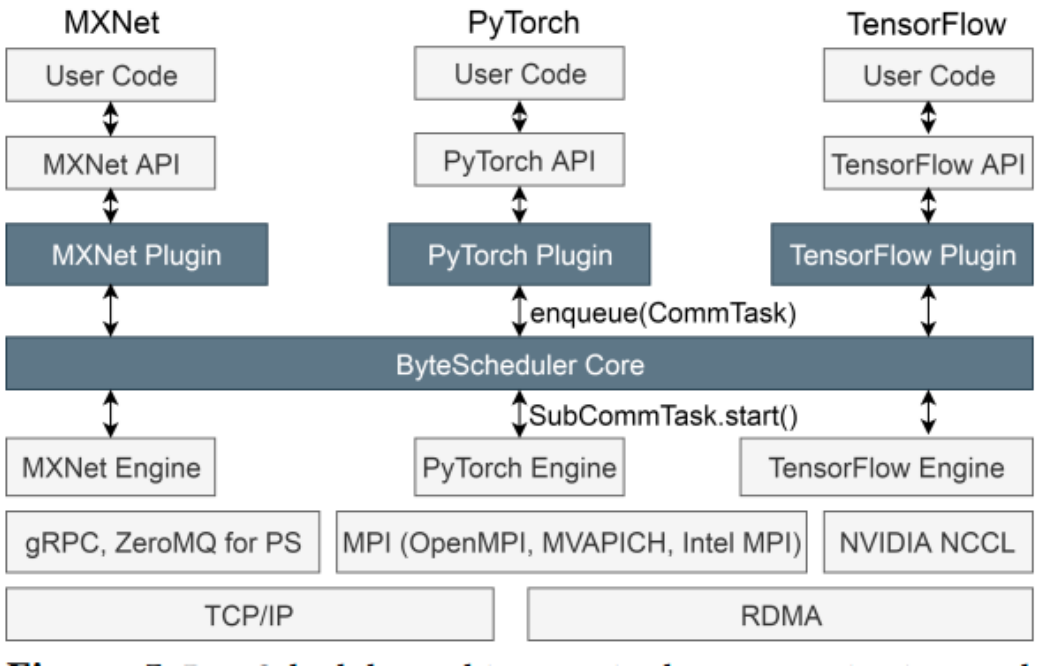
Summarization of 4 papers

Paper 1: (Related to Operating systems and machine learning)

https://dl.acm.org/doi/abs/10.1145/3341301.3359642?casa_token=wO01A7MyRpgAAAAA:psL6QwLqcS5s-orvE2ZSgre-YJZ8w3sYgtQEiDYhuQqUuYm2rkaJRKZoFXMmS9ptwRzBCZdrK3Zj6m4 A Generic Communication Schedular for distributed DNN Trainning Acceleration

This paper is written by professors of HongKong University and ByteDance.

- It mainly introduces a new developed communication scheduler for accrelerating DNN Training. Instead of focusing on the traditional way of accelerate DNN trainning like data parallelism(which would have a significant communication overhead), this approach focus on communication scheduling.
- Problems this paper solved:
    - Some ML framework decide the communication themselves / prevent communications. This scheduler find a generic way to schedule the order of communication in spite of this. (They resolve this by implementing a plugin and core between the API-implementation layer and framework engine). This paper illustrate this idea by the figure below:



  - Previous work doesn't adapt well to wide range of system set-up (They use auto-tuning algorithm based on Bayesian Optimization to help the scheduler adapt to different configurations).
- The scheduler focus on:
  - Scheduling the execution of "layers" in forward propagation and backward propagation of DNN, including scheduling the order of communicaion(For example in figure 1, f1 (forward propagation of layer1) depends on f0(forward propagation of layer 0), So pull0 should have a high priority than pull1. Because f0 must be executed before f1, it is better to finish pull0 as soon as possible).

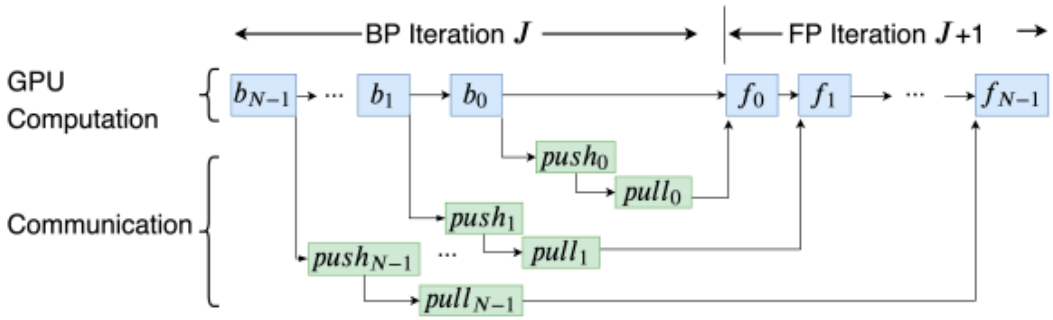
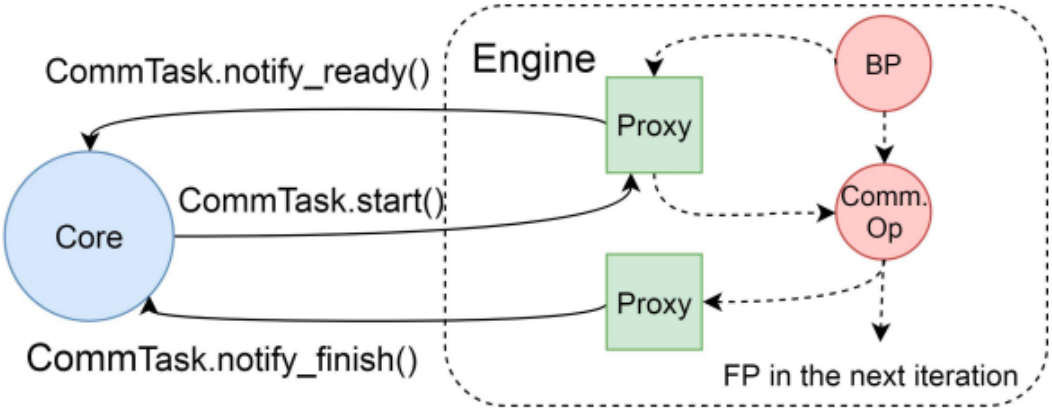
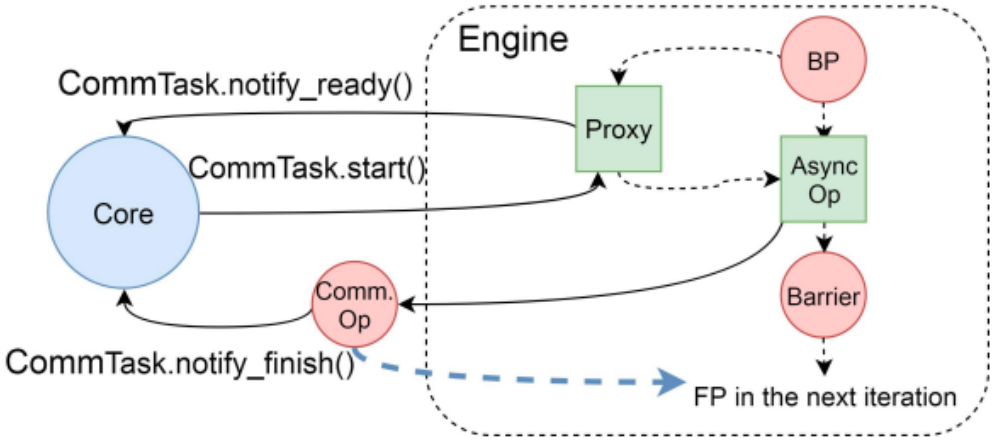


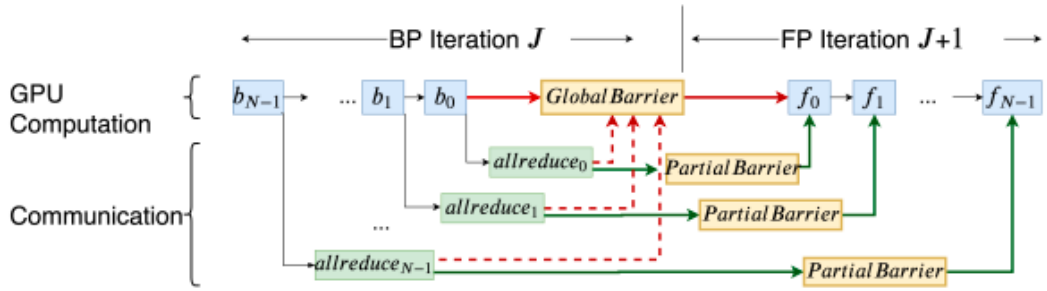
Figure 1. Layer-wise computation and communication in distributed DNN training, *e.g.*, MXNet PS.

  - Tensor partitioning: With smaller size of tensor, it is more efficient to do the pipelining (means bigger speedup). However the partitioning will also bring penatly during run-time environment. Therefore, this is a trade-off when deciding the size of partitioned tensor, so the paper tried to optimized the size of tensor by auto-tuning algorithm based on Bayesian Optimization.
  - Also, it develop some interfaces and a Dependency Proxy(an operation to claim dependencies from/to other operations) to help aid the communication tasks. And the interaction between Core of scheduler and the engine of ML framework would be like:



- Also, for some framework, there exist a global barrier between two iterations (the waiting time of the global barrier for each process should be a significant penalty). To resolve this, this paper use asychronous operation to start the actual communication in the background, and return immediately to let the global barrier pass. This paper use following two figures to show the idea.





- Using this scheduler, they evaluate speedup under different Setups, different DNN models, and different network bandwidths, etc, and found that it achieces up to 196% end-to-end speedup.

Then I read another two papers related to how to using Machine learning to enhance the performance of OS. The first one discusses the optunities and challenges of this study area in a general way. The second one conducted a experiment of applying desicion tree to the process scheduling.

Paper 2:

https://cseweb.ucsd.edu/~yiying/LearnedOS-OSR19.pdf (The authors Yiying Zhang, Yutong Huang are assistant professors in Purdue University)

This paper discusses the possibilities of enhancing the performance of OS using meachine learning. It mainly focuses on the the aspects of OS that can be imporved by ML, in addition to the possibilites and chanllege of applying ML in OS. I gained following things bellow:

- This is a very new area of study and exist few significant outcome.
- It is very possible to help OS get improvement from the bottleneck compared to the previous methods using fixed algorithms. It will be more specific and faster according to the specific use of users.
- The possible usages can be:
    - Some long-standing settings, like certain configurations and policies of OS. This configurations and policies are now complex and general for users. it is possible to adust this configurations and policies to best adapt the hardware and user program (make OS more specific to different users).
    - Adjusting some configurations will affect the application performance and resource utilization, but will rarely affect correctness. such as:
        - space allocation: cache buffer size, virtual memory, swapping, file system...
        - time relating configuration: frequency of interrupting a CPU core, sampling rate of CPU core frequency, cache buffer reflushing
        - cache management
    - ML of improving this jobs can be done off-line, with low frequency.
    - RNN(reuse memory) and Reinforcement learning(take long time(if it is in low frequency, it is OK), but can training with a reward function) can be possible choices of methods.
    - Machine learning model can provide some probabilistic candidates for OS's deterministic tasks. Then OS use the traditional algorithms to compute the final exact answer within this range.
- Challenges:
    - The performance of OS is affected by many factors, how can we evaluate a machine learning model is good or not. (Need to find a persuasive evaluation criterion).
    - Degreee of Finer-grained. Should the model focus on every applications running on the computer or just be applied to some general place. (more specialization means occupying more resources (e.g. CPU time and memory space)), which may not be acceptable by current users.
    - How to ensure the security of this "Learned" Operating system.
- Related work:
    - SageDB: A learned database system which predicts memory access patterns and performs memory prefetching using RNN. (potential to be used in the memory management part issues of OS).
    - Lynx also apply machine learning to perform prefectching from SSDs. It leverages Markov Chains to detect I/O workload partterns and compute transition probabilities between pages.

Paper 3:

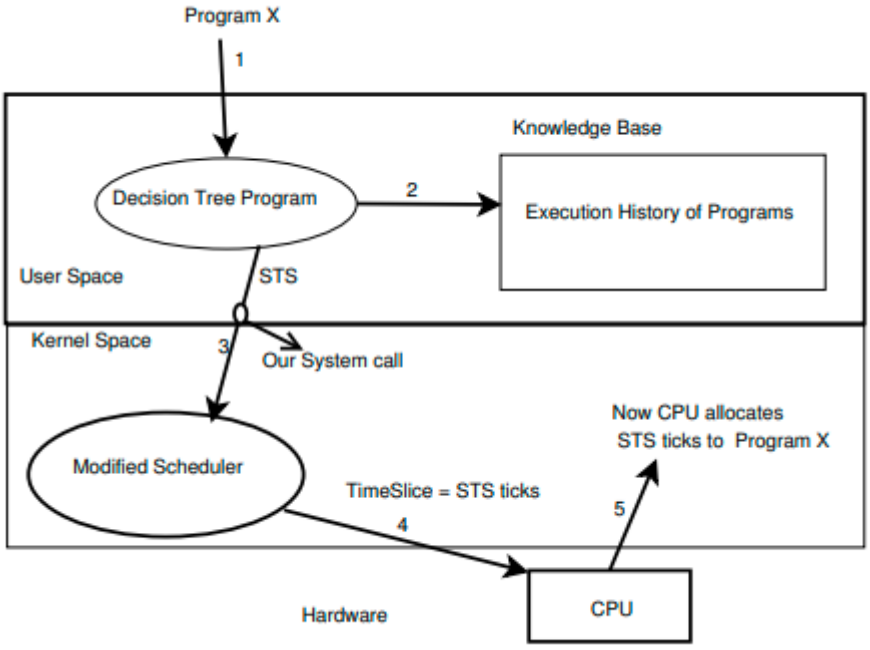
https://ieeexplore.ieee.org/document/4085157 (apply Machine learning to the Process scheduling).

Through this paper, I learned some detailed approach to conduct a experiment about applying ML on OS. (I will try to reproduce this experiment when I get time). This paper using decision Tree and Waikato Environment for Knowledge Analysis to prediction the CPU burst time, in order to minimize the process Turn-around-Time.

Approaches:

- step 1: it modifies the system call and data structure in Linux kernel, (the main variable it controls is **time_slice**, by controlling this variable, the process will get certain ticks of processor time.)
- step 2: The data set is created from the program's run traces(including the dynamic and static characteristic). This execution instances they choose is matrix multiplication, quick-sort, merge-sort, heap sort and a recursice Fibonacci number generator(The execution instance is not that general from my point of view).
- step 3: Choosing of learning features: e.g. hash table size, dynamic linking information size, total size of program, type of input, etc. And the
- step 4: Trainning and Testing (Learning Algorithm: C4.5, and k-NN.) The learning is trying to minimize the Process Turn-around-time.
- step 5: extracting the best features.

Overall model:



Except for the paper about OS, I also read a paper about Distributed and parallel computing. This paper is writen by professors from HongKong University and some researchers from industries. https://dl.acm.org/doi/10.1145/3437801.3441593It is mainly about how to better trian large DNN model using Pipelined Data parallel Approach. Here are some summarization I gained from this research.

Paper 4:

Shortback some of previous works:

- Precious works traditionally used sychronous data parallel to speed up the trainning, which distributed the workload to multiple workers and perform the computation and synchronizes gradients periodiacally. However the synchronization overhead will prevent the linear scalabiltiy (when the scale increase, the overhead is more significant).
- Moreover, the asychronization approach cannot guarantee the convergence of the model well(so it is not common in industry application).
- For some pipeline training, improving computing efficiency and ensuring the convergence of model would bring increasing memory usage (It is a difficult trade-off)

To solve these problem, this paper propose a framework combining data parallelism and pipeline parallelism. There are 3 components: 1. A profiler which inputs a DNN model and profiles execution time, activation sizes and paramemter sizes, etc. 2. A planner which taking profiling results as input and generate an optimized parallelization plan. 3. A runtime, which takes the planner's results and transform the original model into a pipelined parallel one. The authors used this figure to illustrate their idea:

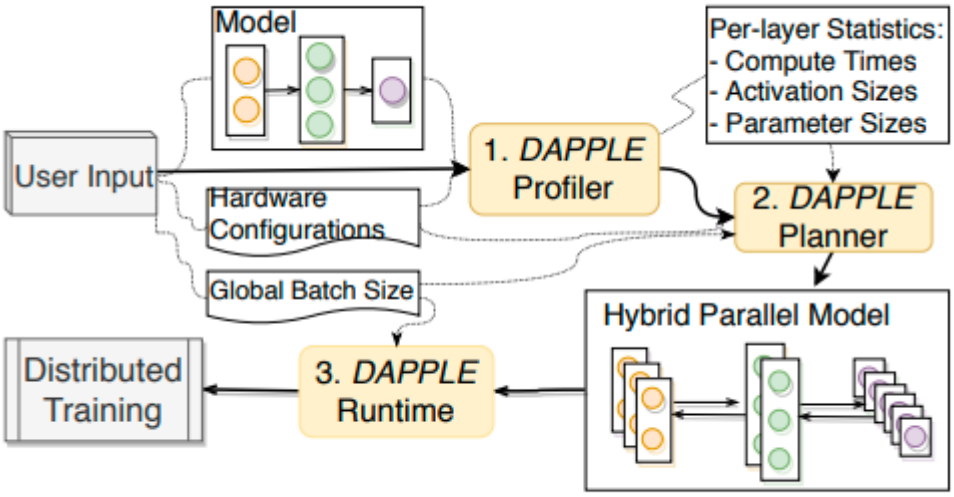


Figure 1. *DAPPLE* framework overview.

Some design Approaches

1. Early backward scheduling (An approach to reduce the memory usage of pipeline)

- Divide the batch of problems into many micro-batches. Then scheduling one forward propagation task of a micro-batch followed by one backward propagation task. So that the memory occupied by the forward propagation task can be freed as soon as possible. This figure compare it with traditional one(Gpipe):

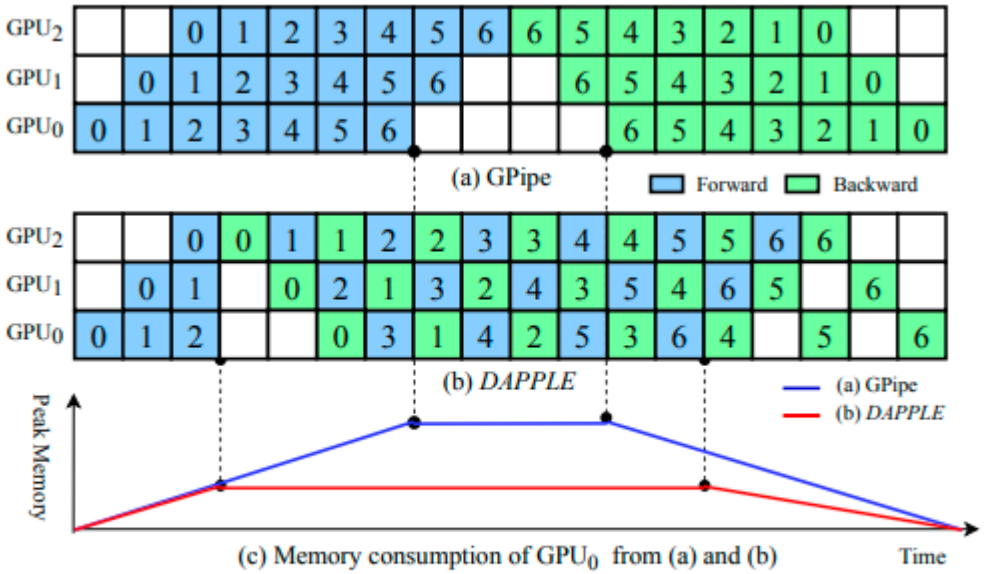


Figure 3. The different scheduling between GPipe(a) and DAPPLE(b) and their memory consumptions.

2. Planner (for giving a parallelism execution plan according to the profiling result)

- It uses dynamic programming to find the optimal partition, replication and placement strategy. And what it wanna optimize is the pipeline latency(the time from start of pipeline trainning iteration to the end). First, this algorithm considers the device assignment (stategies to map the tasks to machines) and it propose 3 strategies: Fresh first, Append first and Scatter first. The principle of these strategies is shown by the figure:

Different stategies get differetn advantages. And the author give an algorithm to find the optimal pivot stage (the page with least bubble overhead) under diffenrent situations (different layers, different number of GPUs and different divice). (I think I haven't understood this algorithm well and need to spend more time in understanding it).

3. Runtime (take user model and planing result as input and transform the model into a pipeline one). It includes the builind of Micro-batch units and scheduling the micro-batch units.

Then the working of this framework would be: The user provides models, problem size,etc, to it. The Planner will give three types of plans: 1. A plan of P:Q(represents two stage pipeline, with the first stage and second stage relicated on P and Q devices). 2. A straight plan (represents pipelines with no replication). 3. A DP plan (data parallel).

Then the runtime will give two scheduling strategies according to the number of succesive forward micro-batches scheduled in the warmup phase(The time period from the start to the pivot stage's(the stage with least bubble overhead) first forward micro-batch). Then using the new Pipeline model to perform the computation.

总结:

目前来说我会对机器学习、深度学习与操作系统和分布式并行式计算的结合比较感兴趣。其中包括两个不同的方向，一个是如何利用机器学习来提升操作系统\分布式系统的的性能。第二个是如何打造更好的操作系统\分布式系统来更高效地进行机器学习。

然后我收集了一些论文，阅读并理解了其中的4篇（其中第四篇（有关分布式并行式计算在DNN中的应用的）对我来说最难，其中planning部分的算法的某些细节还没有完全理解）。除了收获了一些论文内容中专业方面的知识之外，我还收获了以下一些内容：

1. 在收集顶会的论文的时候，感觉有关"如何利用机器学习来提升操作系统\分布式系统的的性能"的研究论文相对较少，并且在我看的第2篇论文当中，也提到了这方面的研究和应用相对于"如何打造更好的操作系统\分布式系统来更高效地进行机器学习"少很多。但它提到了很多这方面研究的可能性与困难，第三篇论文给出了一个具体的这方面的研究应用。给我的印象是这是一个相对较新的，现有研究还较少的领域，对我来说可能会有更多的机遇和挑战，其实对于我来说会更喜欢涉足较新的领域。我会很乐意再读更多有关这方面的论文来认识这个领域的发展的可能性。
2. 对于"如何打造更好的操作系统\分布式系统来更高效地进行机器学习"，我细读的两篇论文分别是有关scheduler，和pipeline和data parallelism。共同点是，它们的目的都是提升DNN的学习效率（同时保证准确率，降低内存使用）。在读论文的过程中会感觉挺感兴趣，并且理解起来也相比于以前我看的其他领域的论文会相对顺畅一些，这应该也与我目前在学校修的操作系统和分布式并行式计算的课程有关系。
3. 在四篇论文中，有三篇是的作者是中国人，分别来自香港大学，普渡大学，和业界（字节跳动和阿里巴巴）。其中来自香港大学的吴川教授参与撰写了第1篇和第4篇论文，这两篇论文都发在很好的会议上，一篇在SOSP一篇在PPoPP，或许我可以尝试发邮件给她申请做她的科研助理。

除此之外，我对于自己的兴趣，基础和擅长也有一些反思：

1. 首先相对于计算机方面的其他领域，我会对system方面有更多的研究兴趣。（是从学校的课程中得出的，我会对操作系统，分布式并行式计算，微处理器系统这几门课有更高的热情）。
2. 另外，在学习和理解的这方面领域的知识（包括学校中学的基础知识，以及读论文）的过程中，会相对与其他领域更加轻松。
3. 当然，目前读的论文数还很少，也只在学校的cluster上做过几个有关并行式计算的实验，还需要读更多的论文才能明白自己是否适合深入做这方面的研究，或者适合做哪个细分领域的研究。
4. 自学过machine learning和deep learning的知识，也学过一些模型和pytorch，但缺乏实际应用的经验。如果要进行"如何利用机器学习来提升操作系统\分布式系统的的性能"方面的研究的话还需要补更多这方面的知识。