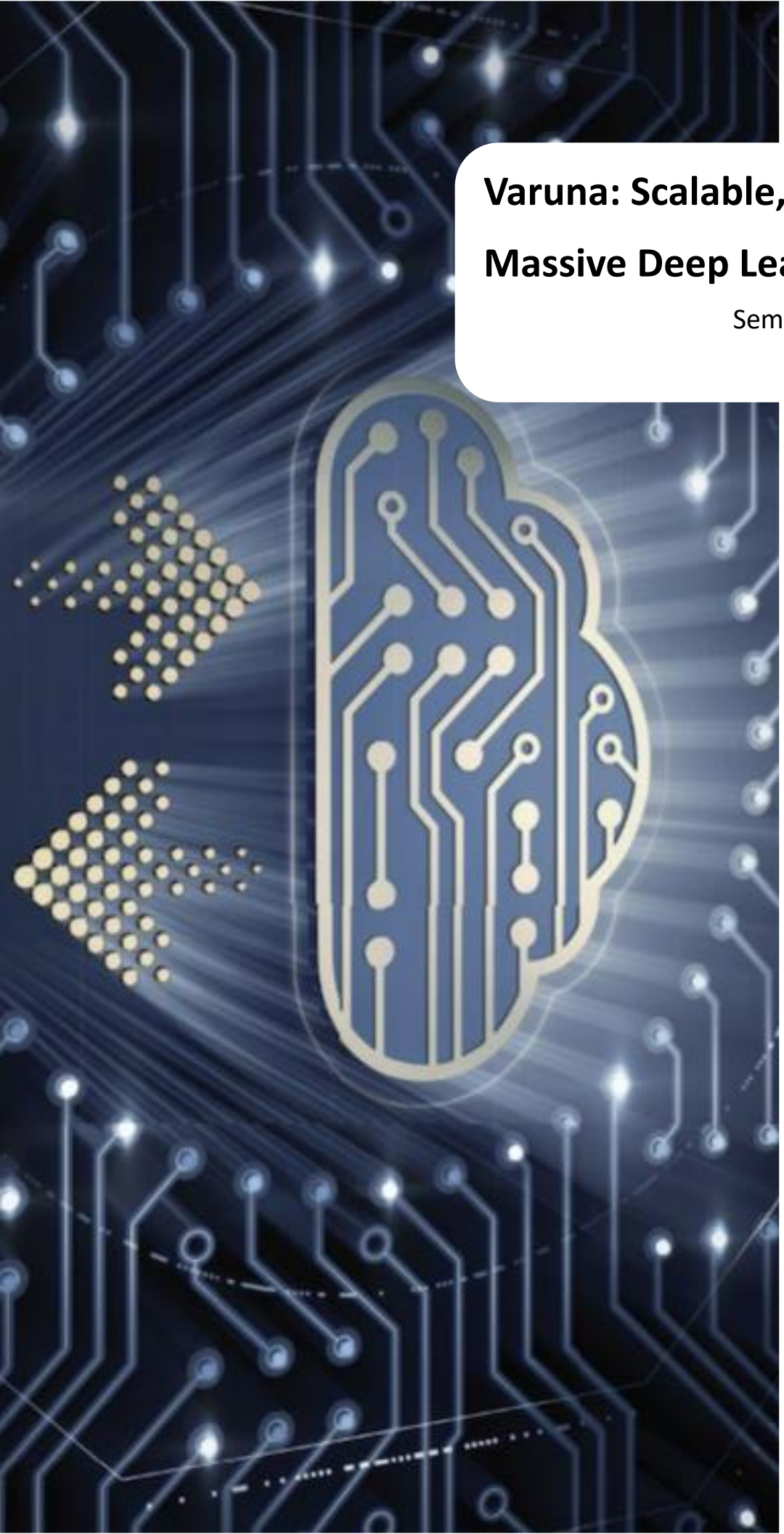

Varuna: Scalable, Low-cost Training of Massive Deep Learning Models

Seminar distributed learning systems

Presented at EuroSys 2022

Paper by

- Sanjith Athlur
- Nitika Saran
- Muthian Sivathanu
- Ramachandran Ramjee
- Nipun Kwatra

Review by Maxime Geminiani Welcklen

Table of Contents

Table of figures

1. Analysis of the paper

The first section of this reviews starts by analyzing the paper itself. Firstly, it details the problem addressed by the paper. Then, it describes how Varuna implements its solution to these concerns.

1.1. Context

Deep learning is present in many forms of modern artificial intelligence, providing good results in computer vision (CV), reinforcement learning (RL), natural language processing (NLP), and much more. The genesis of deep learning can be tracked back to very simple structures such as the perceptron invented in 1958 [1]. This first structure is composed of a list of inputs which are combined with a bias and passed through a simple activation function. During training, the resulting output value can be compared to the expected output to compute an error and adjust the weights of each input in order to achieve a desired behavior.

Many more structures have recently emerged, iterating on one another – such as recurrent neural networks (RNNs) with long short-term memory units (LSTM) [2] or the promising transformers [3] that are showing success in NLP and other types of contextualized tasks such as audio-video processing. However, these advanced mechanisms still revolve around the concept of iterative training via gradient descent.

As the computing structure themselves are becoming more complex, the models are also showing a tendency to become larger over the years.

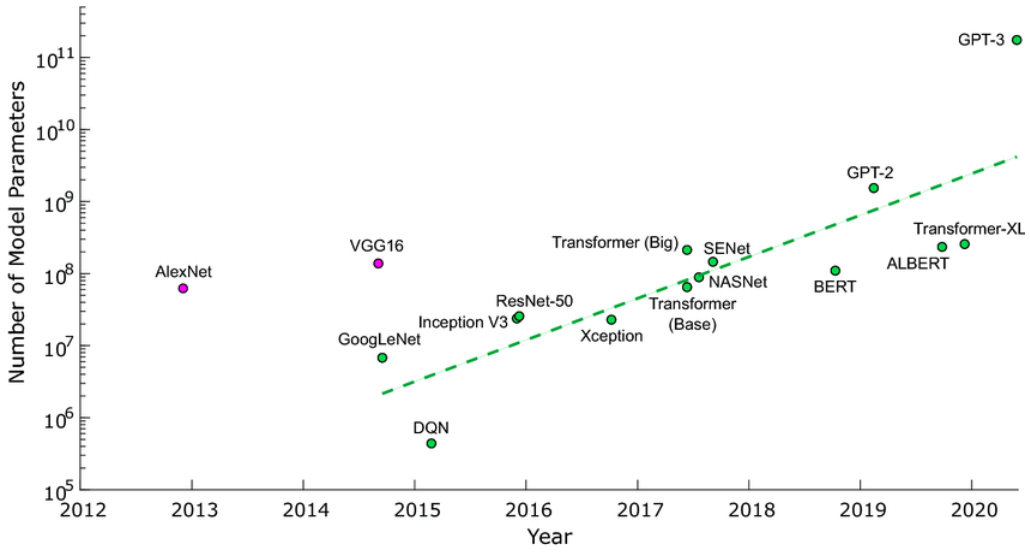

Figure 1 – Growing trend of the count of deep learning's parameters

The size of a deep learning model can be estimated via its number of parameters: the total count of nodes it contains, regardless of the type of nodes. The trend can be viewed in the Figure 1. The parameter count is on constant augmentation and has breached well over the billions.

Meanwhile this parameter count inflation is improving the accuracy and general results of the deep learning models, it has brought new challenges regarding their training. Firstly, the training time increases

with the number of parameters as each of them is called during forward and backward passes. Secondly, each parameter can represent “up to 16 bytes of memory” to be stored during the passes. While the former is an issue for the companies trying to bring solutions in a reasonable time, the latter is a direct problem for the hardware that must run the computation: At a certain point, a single GPU’s memory cannot fit an entire model, let alone run computations and store the relevant values during training.

In order to accelerate the training time of a model, a simple solution is to deploy it over multiple GPUs to share the computation cost between them. Gradient descent can be broken down to multiple sub-parts and unified at the end of an epoch through an all-reduce method. This allows the training to be executed over multiple GPUs, with the connection between the hardware being a possible bottleneck.

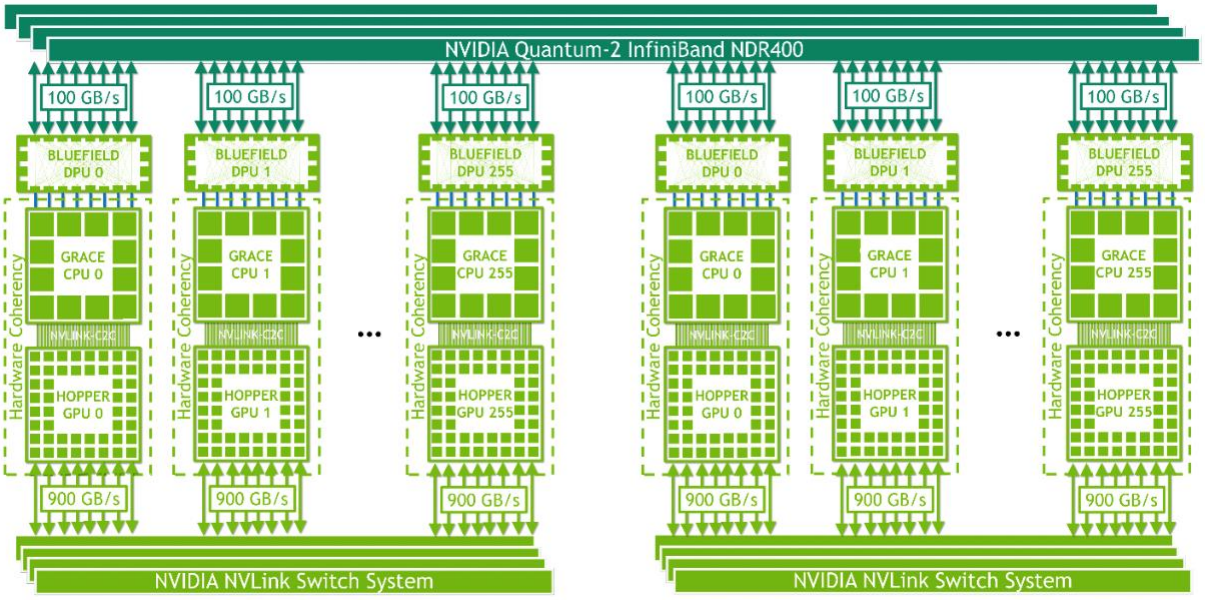

Figure 2 – Overview of a typical setup using NVIDIA’s NV-link and InfiniBand solutions

Large models are often trained on specialized clusters of GPUs connected through very fast networks such as the one displayed in the Figure 2. This solution uses Nvidia’s NVlink for internal communications, and InfiniBand for external communications. Allowing each GPU to run a part of the data through an epoch and agglomerating the partial results is referred to as data parallelism.

However, this does not solve the problem of a model being too large to fit a single GPU. For such gargantuan examples, it is required to split the model itself over multiple hardware.

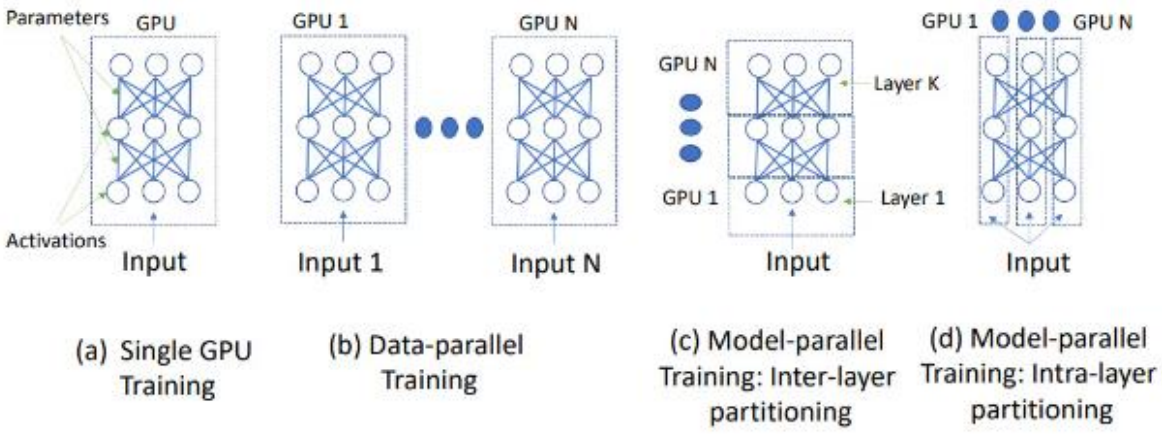

Figure 3 – Different methods of model partitioning

A deep learning model is typically represented as a list of inter-connected layers, each layer composed of a number of parameters. There are two ways of cutting down such a structure displayed in the Figure 3:

- Inter-layer partitioning, where the cut is done at the interface of two layers
- Intra-layer partitioning, where the cut is done vertically through the layers and deploy a number of parameters over a GPU

According to the paper, the state-of-the-art such as Megatron or deepspeed prefer intra-layer to inter-layer partitioning because of the pipeline bubble problem.

The pipeline bubble refers to a phenomenon caused by the synchronous gradient descend algorithm (SGD) – specifically its synchronous component. When a distributed pipeline is processing computation in a strictly sequential manner, then any delay caused by hardware or network failure will be reflected on the following steps of the pipeline – thus creating a bubble of delay propagating through the pipeline. A naïve implementation of inter-layer communication can result in bubbles negatively impacting the computation time of the training.

The specialized clusters are attractive as they offer interconnected GPUs with reliable high-speed networks. They are the cornerstone of training and are seen as a requirement when training large models. However, they are not the perfect solution and can suffer from scalability limits and fragmentation issues. Most importantly, the training cost of a single model are skyrocketing with a paper's estimation of 200k$ for BERT-large, it is possible to imagine models such as GPT-4 with hundreds of billions of parameters to have a training cost become commercially prohibitive. Varuna is all about offering another point of view on the matter.

1.2. Varuna

The goal of Varuna is to offer an alternative to costly GPU hyperclusters for training large-scale deep learning models. It does so by offering an algorithm to leverage low-priority VMs on the cloud.

1.2.1. Deployment

Low-priority VMs are "leftover" virtual machines that are secluded from each other. Contrary to the hyperclusters, they are not provided with a high-speed network communication. However, they are still

powerful GPU machines adequate to deep learning training that are clumped in systems of 1 to 4 GPUs. Their low priority allows them to be picked up at a fraction of their price (up to 5 times reduced) but comes at a price: they can be re-assigned / removed from the low-cost pool after some time. This means that this interesting pool of machines is dynamic, a property that Varuna embraces.

Varuna handles the dynamic pool by automatically adapting the shape of the pipeline to the model's complexity as well as the available resources. It is a trade-off between the advantages offered by the partitioning of the model in reduced training time with the underlying risk of deploying multiple parts of it over unreliable network.

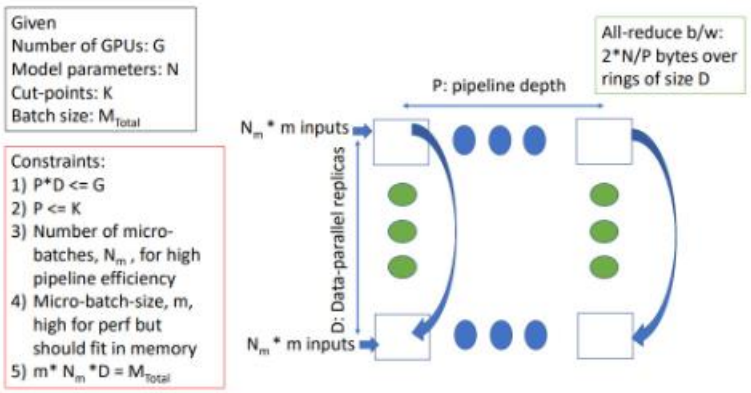

Figure 4 – Structure of deployment by Varuna

Varuna considers its deployment as two dimensional as seen in the Figure 4: A model can be deployed vertically over D instances and horizontally over a depth of P.

D refers to the data-parallelism where every sub-part of the model can be deployed over multiple instances in order to spread the computation and reduce training time. P refers to the inter-layer parallelism where the model is broken down into smaller parts that are more manageable for GPUs with limited memory.

The constraints highlighted in the schema relate to the fact that the deployment is of course limited by the number of GPUs available at time T in the system as well as the flexibility of the model itself to be broken down at predefined cut points. Breaking down the model also increases the vulnerability to networking issues: inter-layer partitioning is conservative on bandwith, with only activations / gradients being communicated between steps, but is prone to the bubble effect. Therefore, the depth is a trade-off that is carefully considered.

Once the model is deployed over the de-facto clustered network of low-priority VMs, Varuna offers an optimization regarding the scheduling of the tasks.

1.2.2. Scheduling

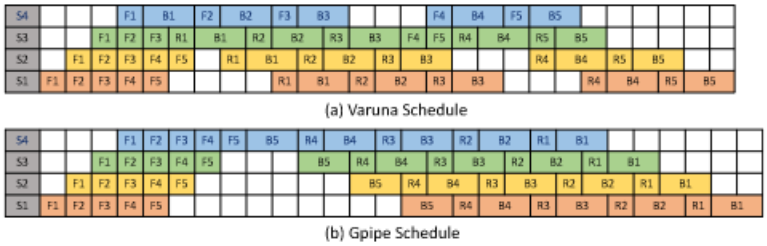

Figure 5 – Comparison of the scheduling algorithm between Gpipe and Varuna

The scheduling algorithm of Varuna does not assume a stable network but tries to reduce the bubble effect to a minimum. The bubble effect arises when a GPU is waiting on the result of a previous step to perform its own computation, and the blocking propagates through the network. In order to keep GPUs busy, Varuna breaks down batches into micro-batches. This allows a batch to be processed in m steps.

The processing of a forward pass is straight forward: a GPU can process a forward pass with the previous activation values and the data. However, a backward pass requires the values of the previous backward passes gradients as well as the activation values of this pass. Storing all the activation values during a forward pass is hardly manageable and the computation of a forward pass itself is less costly than the storage and retrieval of the values in persistent memory. Therefore, a "recomputation" function is preferred during the backward pass.

As shown in the Figure 5, Varuna uses the micro-batches to interleave the forward and backward computations across the network. Forward computations are less affected by jitters as backward ones. Instead of executing all forward and backward computation sequentially, Varuna prioritizes the recompute and backward computation before the forward ones. This allows the GPUs to receive their backward computation task faster than with traditional scheduling algorithms. This also means that in case of delay of a backward computation, there is more chance of having a forward pass to execute in order to keep the hardware working and soften the cost of the delay.

1.2.3. Job morphing

The analysis of the network is done by querying the cloud systems to obtain a list of candidates for the model to run on. Form this list of candidates, a unique profiling step occurs in order to obtain metrics used by the scheduler. These metrics are chosen to be agnostic of the number of GPUs that the model runs on – this allows the scheduler to re-use the values of the metric on subsequent structure of the networks after pre-emption or re-allocation of the pool, thus dealing with the dynamic size of the pool of machines.

Parameter	Description
$F_i(m)$	Forward-pass compute-time for C_i
$B_i(m)$	Backward-pass compute-time for C_i
$Act^i_{intra}(m)$	Latency (same node) to send activations of C_i
$Grad^i_{intra}(m)$	Latency (same node) to send gradients of C_i
Act^i_{inter} (m)	Latency (cross-node) to send activations of C_i
$Grad^i_{inter}$ (m)	Latency (cross-node) to send gradients of C_i
$AR_i(D)$	Gradient Allreduce time for C_i on ring size D

Figure 6 – Metrics estimated by the profiler

The profiling step is executed over a few micro-batches and yields the metrics shown in the Figure 9. This allows the scheduler to anticipate the computation time of the model over the GPUs – minus the network jitters that are unpredictable in nature – and plan an efficient training schedule from the start. The planning is helped by a simulator of the system that is fed by these estimations.

A checkpointing in the form of a constant heartbeat monitoring is constantly established. This provides a view of the network state at all times that allows Varuna to adapt to shrinkages of the GPUs due to pre-emption. New machines can also be requested following a re-allocation algorithm that is run at regular intervals as to not fall under the minimum number of GPUs that would make the model too large to run into. This can be mitigated by further reducing the size of micro-batches to an extent.

This planning, combined with the interleaving of the forward and backward passes, allows Varuna to be efficient over un-specialized networks.

1.2.4. Results

This section discusses the results of Varuna compared to the state-of-the-art algorithms use for large-scale deep learning training.

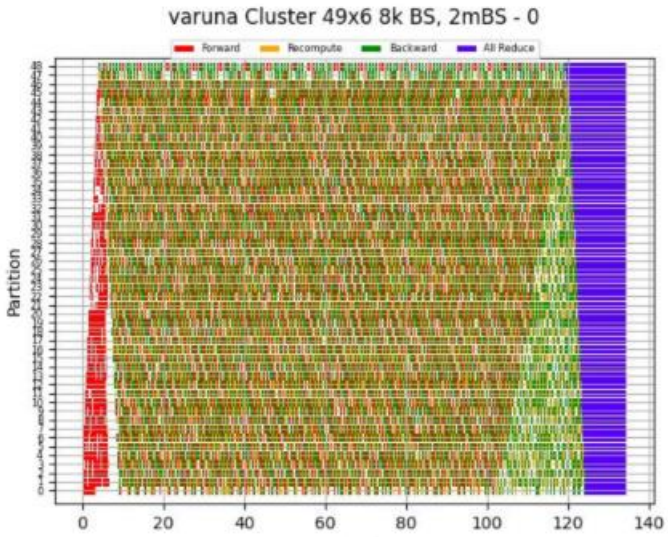

Figure 7 – Varuna checkpoint of the execution over the GPU systems

The Figure 7 highlights the effectiveness of the inter-leaving scheduling algorithm: Aside from the block of initial forward passes and the final block of all-reduce, all the machines are interleaving forward and backward (with recompute) passes over the full duration of the training. Very few bubble effects can be distinguished from this example, excepted for the last GPUs at the end of the training due to the lack of forward passes to compute.

Model	Config ($P \times D$)	Minibatch time (s) Estimated	Actual
8.3B	36x3	142.8	140.3
8.3B	36x2	198.7	201.3
8.3B	36x1	368.3	390
8.3B	24x4	144.7	149.9
8.3B	24x2	272.4	280.4
8.3B	18x6	139.6	139.1
8.3B	18x4	202.6	203.1
8.3B	18x3	266.6	263.8
2.5B	27x2	115.7	116.8
2.5B	18x3	92.6	96.6
2.5B	9x7	68.9	70.1
2.5B	6x10	77.5	75.2

*Figure 8 – Results of Varuna following multiple P*D configurations*

The Figure 8 shows the results of Varuna compared to the estimated results by the profiler-simulator pair. The estimated time are very consistent with the observed values, proving the pair to offer a reliable estimation for the scheduler to rely on.

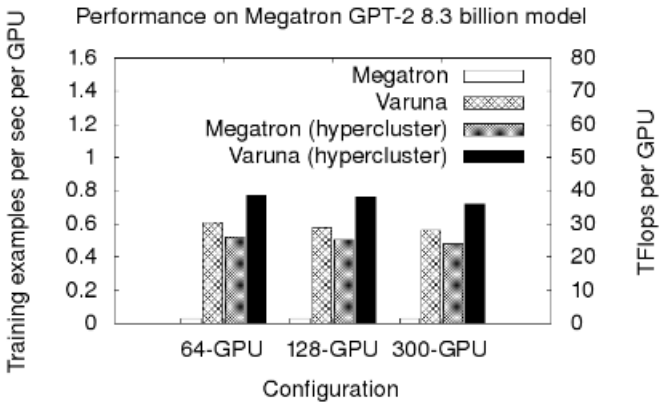

Figure 9 – Comparison of the training time between Varuna and the state-of-the-art algorithms

Lastly, the Figure 9 shows the efficiency of Varuna in term of computation time. It beats Megatron, a direct competitor, over both its hyperclusters and non-specialized deployment. Although the non-specialized deployment of Megatron (white bar) can be considered somewhat disingenuous as Megatron is made to run on a hyperclusters deployment, it is interesting to see that Varuna outperforms it over its low-priority deployment and that it scales even better on an hyperclusters.

2. Critical discussion

This section proposes a point of view of the paper's strengths and weaknesses.

2.1. Strengths

Varuna's acceptance to the Eurosys conference shows that it offers interesting perspectives to the field. This section highlights the quality or innovations of the paper.

2.1.1. Efficiency

The first strength of Varuna is natural: it greatly outperforms the competition's state-of-the-art algorithms, both on a low-priority and hyperclusters setup. This allows a team without specialized cluster access to train a large model and opens up the door of large-scale deep learning to a wider range of users. Furthermore, it improves the training time of large-scale models – and thus the cost of said models – on standard GPU setups. Large scale models are bringing more accuracy and skillset to the field and training them with a reasonable amount of resources is a key to making them commercially available to the masses. To make it better, an improvement of the training time of a model is worth multiple time its gain, as models can undergo multiple training sessions during which the hyper-parameters are carefully studied to fine-tune the model to a particular task.

2.1.2. Scheduling algorithm

An important strength of Varuna is its scheduling algorithm. The interleaving of the forward and backward passes is an efficient solution that allows it to deal with network jitter and outperform even on GPU cluster setups. It leaves very little downtime for the system – a problem that is continuously being monitored on other deep learning scheduling paper, as GPU clusters can have very low total utilization metrics, where half the cluster can sit Idle due to downtime in the training and miss-schedules. Moreover, the paper proves that inter-layer partitioning is viable and efficient whereas intra-layer was the norm / preferred option at the time of publication.

2.1.3. Innovations

Varuna brings two innovations to the state-of-the-art (SotA). Firstly, it is able to use inter-layer partitioning very efficiently, whereas SotA were focusing on intra-layer partitioning. Secondly, it proves that low-priority VMs can be efficiently used instead of GPU clusters. This is a bold statement that would probably be doubted at first.

2.2. Weaknesses

As thorough as it can be, a scientific paper will have parts that could have gone deeper one way or another. This section discusses some weaknesses of the paper that could be interesting to investigate.

2.2.1. Intrusivity

Varuna highlights its non-intrusiveness multiple times. However, when investigating the documentation, it can be noted that Varuna will work by annotating an existing model.

```
from varuna import CutPoint

class SampleModel(nn.Module):
  def __init__(...):
    ....
    self.cutpoints = [CutPoint() for i in range(num_cutpoints)]
    ....

  def forward(input...):
    input = self.some_operation(input)
    input = self.cutpoints[0](input)     # marked as a potential stage boundary
    input = self.some_other_operation(input)
    ....
    for i in range(sub_modules):
      x = sub_module_i(input, ...)
      x = self.cutpoints[i+1](x)        # each cutpoint instance should be used only once in a model
    ....
```

Figure 10 – Example of setup for a cutpoint in python

For example, cut points are pre-defined by the developer when adding Varuna as shown in the Figure 10. Moreover, the paper states that Varuna also modifies the core libraries to solve some problems related to partitioning, such as vectors that are shared across the models: *"We make minor modifications to the PyTorch library to support a profiling mode, during which each created Tensor is marked with a cut-point number to which it belongs"*.

This means that Varuna requires the model's source code to be adapted in order to work and requires a modification of some core libraries. This is very intrusive and forces the developer to adapt to the scheduler used to train a model. This also requires maintenance: A team that wants to train its model using Varuna must adapt the source code to this specific algorithm and to make sure that the changes to the libraries are not bringing bugs or failures or repair them once they appear.

2.2.2. Low-priority scarcity and unspecialized network overloads

The whole premise that Varuna is based on is to leverage low-priority VMs to have access to very cheap resources and cut down on cost. However, if Varuna was widely used and most GPT-type models were using low-priority resources, then those resources would not be low-priority anymore. While the paper leverages low-cost VMs, the inherent affordability of said VMs stems from their limited usability. Paradoxically, the increase demand of a wide adoption of Varuna may lead to a surge in cost, thereby negating its premise.

The widespread adoption of low-cost VMs, if undertaken en masse, could potentially exert a significant impact on networks. The highly specialized hyperclusters have in-built massive high-speed networking that allows them to bear the load of GPU communications, and the massive use of low-cost VMs could have a strain on those unspecialized networks. This could prompt a reaction from cloud provider to refrain training large scale deep learning models on unspecialized machines or lead to an increase in prices to counterbalance the situation.

2.2.3. Security concerns

GPU clusters are often bought and owned by one or multiple tenants. This ensures sovereignty and security over the data that these large model trains on. The very nature of Varuna's low priority VM and

the constant GPU communication of gradients over un-sovereign or even unprotected networks could be a very serious concern to any model that handles sensitive data. This is not an issue for a large language model (LLM) that use public data, but this is a major concern for a non-negligible part of the field (banking, health, confidential company data…).

2.2.4. GPU heterogeneity

Varuna states that a single round of profiling is necessary to extract metrics related to the time taken for forward pass, backward pass, network latency, etc… This entirely relies on the assumption of a homogeneous group of low-priority GPUs. However, it is not wild to assume that a group of low-priority VMs could be heterogeneous in the sense that multiple type of GPUs can be leveraged at once. Multiple types of GPUs could also be allocated after the single-shot profiling occurs. For this reason, the single-shot profiling is shown as a strength but can only be done on a very specific setup where every GPU allocated in the cloud is the same. In all of the paper's tests shown in results, the same kind of 1-GPU and 4-GPU nodes are used as low priority resource. Varuna could need to be adapted to work with multiple type of GPUs as the single-shot profiling would not hold under this setup, and the scheduler could have trouble adapting to the system it runs on.

2.2.5. Absence of metrics in the results

Whereas the paper's introduction and results heavily states that the costs were ultimately reduced by 4-5 times, there is no study presenting the cost as a metric. The only comparisons between Varuna and its competitor show the accuracy as a variable of the model used. It would be interesting to see the cost as a variable of the model and structure of clusters / low-priority VMs used.

Similarly, the financial cost is not the only one that is to be considered. More and more studies are showing the ecological cost of training large-scale deep learning models. By using unspecialized VMs and communicating over unspecialized networks, the overall computation efficiency of the training could be reduced and this could have a negative impact on the harmful emissions of the training. While highly specialized clusters are by definitions made for the task and optimized for it, low-cost VMs could include older or faulty hardware.

2.2.6. Continuity of work

Varuna's code has been open sourced following the paper publication. However, the repository [4] is abandoned since the end of 2022, hinting that the solution has not been adopted seriously at Microsoft. Furthermore, one of the github action (#17) titled "FAIR VIOLATION - ACADEMIC DISHONESTY!!" and renamed to "oops" in February 2023 can raise an eyebrow about the seriousness of the work done.

3. Conclusion

The conclusion presents some avenues of improvement following the previously established strengths and weaknesses of Varuna and finishes on a wrap-up and personal notes over the paper.

3.1. Avenues of improvement

This section proposes some improvements or interesting discussions about changes that could be beneficial to Varuna.

3.1.1. Analysis of model's structural differences

As discussed in the review's introduction, modern deep learning structure employ complex nodes such as LSTM or transformers. Even more classical structures such as convolutional networks (CNNs) will often have an unbalance of computation between the different steps of the model. However, Varuna relies on users' annotation of cut points. However, a user's annotation could be misplaced or unoptimized regarding the model's architecture. Therefore, it could be interesting to have more strategically placed cuts for the inter-layer partitioning, either by having a predefined set of loads for different architectures, or by automatically partitioning entirely from the profiling where computation time can be checked at different steps of the model.

3.1.2. Vendor lock-in

As a Microsoft research paper, Varuna uses Azure's APIs to query the could availability and setup low-priority VMs. This leads to a vendor lock-in: any Varuna user would be locked with the azure environment. There are several tools that allow infrastructure as code (IAC) and easy over-the-cloud deployment such as terraform and Kubernetes that could help Varuna be more cloud agnostic. The interface for the cloud Api's queries could then simply be swapped depending on the user's preferences, or a cloud-hybrid setup could be put in place to obtain the lowest cost amongst the VMs over multiple domains.

3.2. Final word

Varuna is a well-written paper that take the time to detail the underlying mechanisms such as data and inter/intra-layer parallelisms that are at the core of the concerns it addresses. It proposes an elegant and simple inter-leaving solution for its scheduler and proves that inter-layer can be used efficiently over any network. It also successfully shows that training deep learning models can be done on off-the-shelf cloud GPUs and don't necessarily require a dedicated GPU cluster to work. In a time where models grow larger, it is important to have adapted training methodologies for the hardware to follow the innovations of the fields.

However, although the objective of the paper seems to be to reduce costs, it focuses on training time during its analysis. It would have been interesting to highlight and study the training cost of the models over multiple runs. Varuna is a bit of a two-in-one paper where the scheduling algorithms is a very important part of the results it obtains in term of training time. However, the premise of Varuna is to make a training work over low-priority VMs, a prospect that can be lost or over-shadowed by the very favorable results of the scheduler. This especially shines with the fact that Varuna outperforms Megatron on a regular GPU cluster.

Varuna is an interesting paper but does not seem to appear as influential as BERT or attention is all have been in their time – it offers a very interesting and innovative perspective on deep learning training but does not seem to have gained much traction in term of use. Deep learning schedulers are being heavily studied and new propositions emerge regularly, although a lot of them are focused on cloud-based deployment (such as Kubernetes) to be more agnostic of their environment, which Varuna lacks as it is heavily specialized in the Azure environment. The properties of Varuna of running on low-priority VMs can also be considered as a rebuttal because of security and sovereignty issues and is ultimately not suited for every needs on the market of large deep learning models. This could be a saving grace as using low-priority VMs could paradoxically lead to the downfall of low-priority VMs due to the increased demand.

4. References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review,* 1958.

[2] S. Hochreiter, "Long Short-term Memory," *Neural Computation,* 1997.

[3] N. S. N. P. J. U. L. J. A. N. G. L. K. I. P. Ashish Vaswani, "Attention Is All You Need," *NeurIPS.*

[4] Microsoft, "Microsoft/Varuna," 2019. [Online]. Available: https://github.com/microsoft/varuna.