Limits to Scale-Out

for Training Language Models

Limits to Scale-Out for Training Language Models

Natural language processing has revolutionized how data is consumed, meaning that computational demand has skyrocketed. Companies in every industry are using graphics processing unit (GPU) clusters to keep up. But is this really the best solution?

Cerebras Systems has taken a deep dive into the difficulties of producing NLP models and how those challenges may be affecting the rate at which models can be trained and deployed.

Natural Language Processing — A Game Changer for Businesses

Even before the internet, the world was awash in information — mostly in the form of the written word. Then, with the advent of the internet and big data, the flow rate of information and the multiplicity of portals to access it exploded. The amount of information suddenly at our fingertips was too much to be sorted by any one person or any group of people. We needed to invent ways to extract meaning, and we did, by developing deep learning for natural languages.

Finding something on the internet generally meant asking compilers of directories (of which Yahoo was the leader) for lists of potentially relevant websites. The results were less than satisfactory. Google, founded in 1998, created the first fundamentally accurate and efficient, if imperfect, way to find what was most useful in the flood of information that was the internet. This success was totally algorithmic and relied on simple, mathematically-based ideas to characterize the data's significance. This approach presaged the current revolution in artificial intelligence in which data, rather than direct human insight, trains models (that we cleverly architect — we are not obsolete just yet). Now these models can perform many tasks at a level that humans cannot hope to match.

BERT (Bidirectional Encoder Representations from Transformers) is a natural language processing (NLP) model that understands the meaning of words in the context of sentences and paragraphs. Appearing in 1998, BERT achieved state-of-the-art results on various NLP tests. It was quickly established that BERT could extract meaning not only from text, but also from sequence data such as numeric time series, nucleic acid sequences or the amino acid sequences of proteins. Networks similar to BERT have been put to use in finance, geology, genomics and patent classification, and have even proven successful at handling information sources that span multiple languages.

While working on the development of applications for narrow fields, an important discovery was made.

> Training a domain-specific BERT model on a restricted training set **yields better results for that domain than a generic language model** does when trained on a wider, less focused language corpus.

This discovery implies that we will want to train and retrain numerous BERT-based models as we find new applications, acquire better training data and demand better performance. That creates a large demand for computation.

Let's look at the difficulties of producing NLP models and how those challenges may impact the extent to which customized models trained and retrained for specific uses can be effectively produced and deployed.

Addressing High Training Cost in Large NLP Models

We need to adapt and retrain large NLP models quickly and at a low cost. But training a modern NLP model is hard, and is only getting harder as models grow — and grow they do. A more powerful larger version of BERT, known as BERTLARGE, has the potential to provide better results, but training it on a new corpus may be impossible. Potential users report that:

> Despite our best efforts to use $BERT_{LARGE}$ we used only $BERT_{BASE}$ due to the computational complexity of $BERT_{LARGE}$.

> Such high model complexity calls for expensive computational resources and extremely long training time.

Large BERT models can be trained on a single GPU, but even with the latest and fastest GPU, the training may take days. To speed things up, clusters of GPUs can be used instead. However, while the use of multiple GPUs in parallel does reduce training time, it also adds to the hardware, energy and administration costs.

Nvidia is one of many technology companies that has worked on training a BERT model for their specific needs. The results reported by Nvidia for training a BERT model in the MLPerf benchmarks[1] are shown in Figure 1.

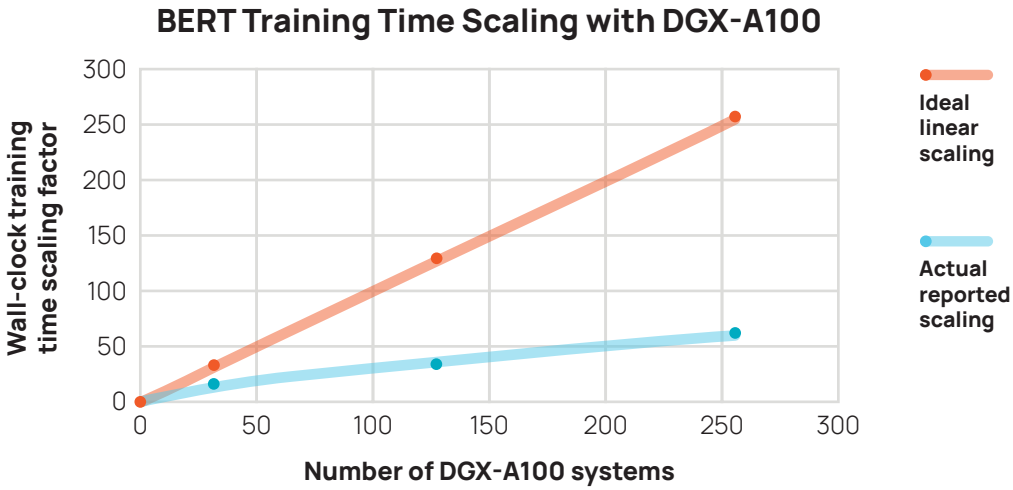
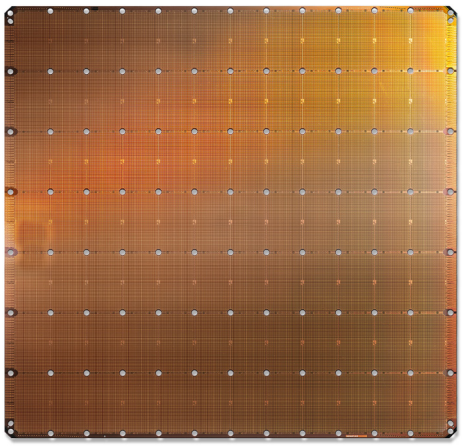


Figure 1. Performance scaling of BERT training on DGX-A100 clusters, per MLPerf

The performance scaling when training BERT on Nvidia A100 clusters ranging from one DGX-A100 server (with 8 GPUs each) to many racks of DGXes is shown (the blue curve); it lies well below the orange line that shows perfect scaling. The performance achieved by adding GPUs grows, but it grows more slowly than the perfect speedup, shown in orange. Note that the slope of the blue achieved performance curve drops as the number of GPUs increases. Thus, the performance scaling is not linear. Rather, the best fit to the data shows that with very good approximation:

$$time = 228.3\ p^{-.7392}$$

1 MLPerf results at ML Commons as of July, 2020.



In a situation of linear scaling, the exponent in this formula would be -1. Here, it is only about -3/4, revealing the considerable loss of efficiency that occurs when using very large clusters of DGX-A100s.

It is quite interesting to try to understand why this happens. An Nvidia blog[2] details the approach taken to achieve scaling to the very highly parallel multi-GPU systems used.

The general approach to using a cluster for training is called data parallelism. In a simple data-parallel implementation, each GPU has its own copy of the parameters of the neural network to be trained. In a single step, each GPU is given its own minibatch of training data and, independently of the others, it computes an update to the model (as a gradient of the loss on the minibatch, scaled by the learning rate, for one possibility). Each of these updates is different. To prevent each GPU from subsequently going its own way, all the individual updates are added together and then the mean of these updates is shared back to all of the GPUs. This allows them to independently update their models using identical mean updates. This way all the GPUs move on to the next step with the same model.

Naive data parallelism uses a constant batch size b on each GPU, regardless of p, the number of GPUs employed. Thus, each step of updating the model uses an effective batch size of $B = b\ p$, which will grow into the thousands on the full-scale systems used by Nvidia.

> This raises the question of what effect large batches have on neural network training. Do they help or not? Unfortunately, recent work on the effect of batch size on the number of gradient descent steps **shows that very large batches (B) are completely ineffective.**

While the number of required model updates falls as $O(1/B)$ when the batch size is small, at a fairly modest batch size the number of update steps stabilizes, and thereafter shows no further reduction.[3] This is quite intuitive. Gradient descent methods are known to be slow by nature. They take a long path, following the curved, slowly descending contours of the objective function. When the batch size gets beyond the point of providing a good gradient approximation to the gradient of the mean loss on the whole training set, further increases just mean more time spent with no added benefit.

---

2 Optimizing NVIDIA AI Performance for MLPerf v0.7 Training (Nvidia blog.)

3 C. Shallue, J. Lee , J. Antognini, J. Sohl-Dickstein, R. Frostig, G. Dahl. Measuring the Effects of Data Parallelism on Neural Network Training. Journal of Machine Learning Research 20 (2019).
N. Lee, T. Ajanthan, P. Torr, M. Jaggi. Understanding the effects of data parallelism and sparsity on neural network training. ICLR 2021.

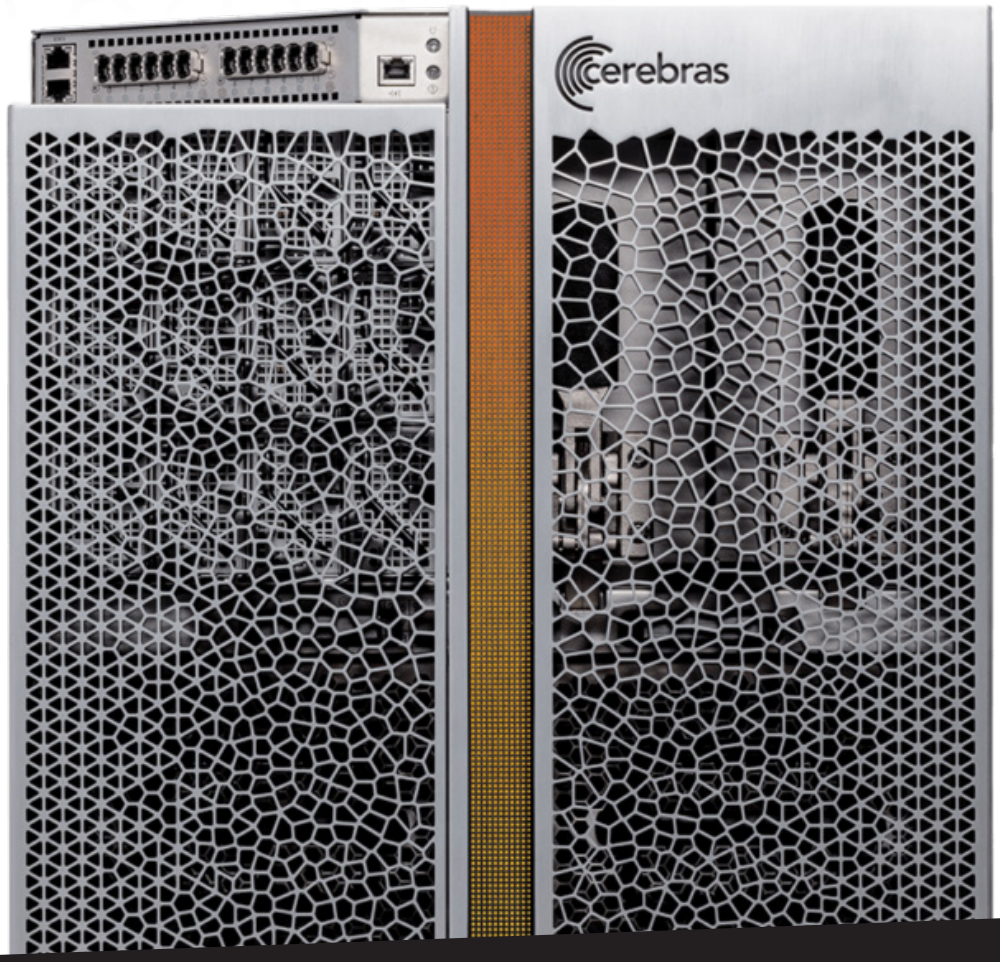
This explains why Nvidia does not take a naive approach. Instead, they use small batches per GPU and per optimization or model update step. This allows the local gradient computation to be relatively fast. Unfortunately, the model update then becomes a bottleneck. Nvidia fully parallelizes the required gradient summation and the model update, finally using an MPI collective communication all-gather operation to restore identical updated models across the cluster of GPUs. The cost of these collectives very likely contributes to the loss of scaling. And at large scale, even with very small local batches, updates may have pushed beyond the point of $O(1/B)$ scaling in step count to accuracy, resulting in another driver of scaling loss.

Nvidia did more to achieve their results. For BERT, they developed new CUDA kernels for multiheaded attention that provide several speed improvements, notably by fusing work into one kernel to reduce the frequency of host-GPU iteration. They also developed new technology to record and replay host actions rather than using the host at every gradient descent step, as this was mandated by the small batch size and parallel optimization algorithm that reduces the runtime of each optimization step enough that the GPU would be a bottleneck without this. The effect is that runtime does scale. But as we saw above, it scales sublinearly.

Conclusion

To achieve the needed runtimes for training large NLP models, the use of large GPU clusters is feasible, but there are downsides. Per the Nvidia data, to achieve a 60-fold reduction on the single DGX, 8 GPU runtime (from 49 to 0.8 minutes) required a 2048/8 = 256-fold increase in the number of GPUs with corresponding increases in size, cost and power. A system of that size has a nominal power consumption of over three-fourths of a megawatt. Moreover, to achieve that result requires not only access to the whole cluster, but failure-free operation of the cluster over the course of the run, and, perhaps most importantly, significant tuning of the implementation. Indeed, the 2021 ICLR paper by Lee, et al, cited above, details the extensive effort that was needed to tune hyperparameters first to find settings for which convergence is achieved, and then to obtain optimal convergence speed.

The challenges to scaling training on GPU clusters can be overcome with the Cerebras CS-2, powered by the Wafer-Scale Engine. **Contact us at cerebras.net/get-demo to see how.**



cerebras

Cerebras Systems is revolutionizing compute for Deep Learning with the CS-2 powered by the Wafer Scale Engine. The Wafer Scale Engine delivers more compute, more memory, and more communication bandwidth for artificial intelligence research at previously-impossible speeds and scale. Pioneering computer architects, computer scientists, and deep learning researchers have come together to build a new class of computer system that accelerates AI by orders of magnitude beyond the current state of the art.

CEREBRAS SYSTEMS, INC. | 1237 E ARQUES AVE, SUNNYVALE, CA 94085 USA | **CEREBRAS.NET**

cerebras