Gee PS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server

Wang Jian

2020 年 11 月 17 日

目录

1 Author

https://cuihenggang.github.io/

2 Notes on a paper:

GeePS is a smart data manager (data migration/movement coordinator) for a very large distributed deep learning training on GPUs. The paper addressed a real need and showed how a problem of distributed learning with GPUs can be solved.

2.1 Tree primary contributions:

1. it describes the first GPU-specialized parameter server design and the changes needed to achieve efficient data-parallel multimachine deep learning with GPUs.

2. it reports on large-scale experiments showing that GeePS indeed supports scalable data parallel execution via a parameter server, in contrast to previous expectations [7].

3. it introduces new parameter server support for enabling such data-parallel deep learning on GPUs even when models are too big to fit in GPU memory, by explicitly managing GPU memory as a cache for parameters and intermediate layer state

2.2 Background

1. Deep learning with GPUs

2. Parallel ML using PS

3. **MXNet uses ParameterServer which is CPU-based parameter server**

2.3 Design

As mentioned earlier, GeePS shards the parameters across the machines. The parameters are stored as an array of fixed size called rows and each row is associated with a key. Rows are bundled together into tables and each table stores information like data age, which denotes how many times the values in the table has updated and requires a refresh.

In addition to this each instance has a parameter cache, local data and access buffer pool. Parameter cache(write back) caches all parameters of the NN in GPU(can be overflowed to CPU memory). This reduces the communication overhead during an iteration. Local data (also stored in

GPU & CPU) are the additional memory needed for intermediate states. This is local to the machine and not shared across the cluster. Access buffer pool is used when application makes read and update calls. GeePS first allocates a buffer in GPU memory and copies the data to it before letting application to read or write to that buffer. This ensures data consistency.

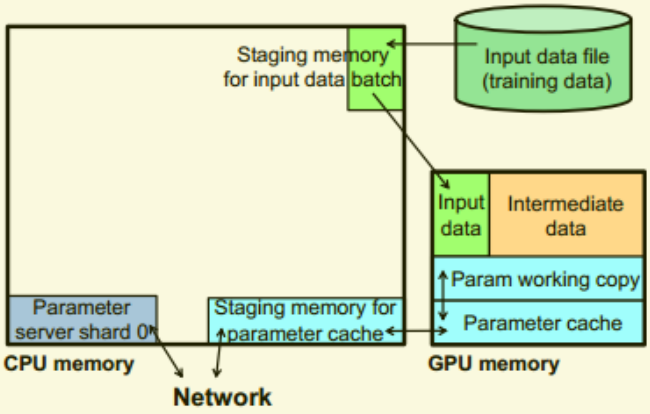

Figure 6. Parameter cache in GPU memory. In addition to the movement of the parameter cache box from CPU memory to GPU memory, this illustration differs from Figure 5 in that the associated staging memory is now inside the parameter server library. It is used for staging updates between the network and the parameter cache, rather than between the parameter cache and the GPU portion of the application.

2.4 Implementation

GeePS prebuilts the indexes based on the details collected during a dry run before the actual training starts. Since NNs computations are repetitive this information is not going to change. During the training iteration, the application reads and updates the parameters on individual machines and at the end GeePS triggers a TableClock. This signals all machines to synchronize the data based on the BSP or SSP policies. The data transfer between machines is done using 3 threads, keeper, pusher and puller. Pusher pushes

updates from the master machines(of a set of parameters) to other machines to cache. Keeper receives this updates and forward it to Puller for updated GPU cache.

The data allocation and reclamation is also done using 2 threads called allocator and reclaimer. These threads synchronize each other with mutex locks on parameter partitions. GeePS avoid using row level fine grained locking. Usually there is only one CPU thread interacting with GeePS and the lock contention is not a big deal.

2.5 Evaluation results

GeePS is evaluated against image classification workloads. It provides 13x improvement in throughput using 16 GPUs. Less than 8% of the time is lost to GPU stalls as compared to CPU based parameter server. GeePS gets same throughput as ProjectAdam, a state of the art CPU based distributed deep learning solution using 108 machines, with only 4 GPUs.