2023-06-28

Agenda

Accelerate JAX models on Intel GPUs via PJRT	Xiao Yu, Google Yiqiang Li, Intel

Xiao Yu and Yiqiang Li presented “Accelerate JAX models on Intel GPUs via PJRT”

Does PJRT support query interface so that the framework can offload computation to different type of device?

There is no query mechanism. For JAX, the whole stable HLO graph is sent to PJRT to compile and run on one Device. So JAX user decides the device for the computation. For TensorFlow, it chooses some of ops to CPU vs. GPU. But in both cases, the stable HLO graph sent to PJRT for compilation is for one device type.

How this looks from end user perspective? Does framework provide unified address space to user?

TF front-end has device type like CPU vs. GPU. Users specify the device type, not the memory. When users copy the tensor from one device to another, users need to copy memory explicitly.

Does the pluggable device support cost model?

There is no cost model. JAX places whole graph on one device. Tensor Flow bridge is responsible to decide which part of graph to pass to XLA. It doesn't have a cost model for that either. The criteria is simple: if the op is supported by XLA, then give to XLA. TF eigen kernel has a cost model but was not used for the decision.

For the AOT compilation, how the compatibility issue of runtime library is considered? If the compilation assumes different runtime library version, it might cause issue.

PJRT leaves the compatibility issues to the PJRT plug-in implementation. The PJRT plug-in needs to have mechanism to ensure the runtime and library being loaded is compatible with the one assumed during compilation. Different Plug-in can choose different compatibility level support.

For the AOT compilation, the compiled graph needs to be serialized, does it support dynamic shape?

As of today, the compiled graph is static shape only as Stable HLO doesn't yet support Dynamic shape.

Is there any mandate on the latency, say when the compiled code issue library call, could the library do some JIT compilation underneath?

To maximum the benefit the AOT compilation, there should be no JIT compilation happens at the runtime. However, it is banned if there is compilation happen in the runtime for best performance. When a module is loaded, usually there some warmup runs, which will allow the library to do the JIT compilation underneath.

For the sparsity support, what are the PJRT API supports?

PJRT needs to pass an annotation to distinguish the input as a sparse tensor and describe the structure.

What are the other PJRT plugin?

Apple also announce metal plugin. AMD and ARM are interested in exploring it.

Is it possible for framework to mix-use two PJRT GPU implementations?

Right now, we don't ban it. But there is no concrete use case. When there is real use case, we can have multiple ways to support it.