Anolis OS 23 intel_ai镜像使用手册

简介

intel_ai镜像的基础OS是Anolis OS 23，镜像中包括了目前最火的AI软件框架TensorFlow 2.12.0和PyTorch 2.0.1，旨在利用阿里云x86实例最新的硬件特性来加速深度神经网络计算。

具体来说：

- TensorFlow是深度学习领域广泛使用的机器学习框架，该框架已基于深度神经网络库oneDNN所提供的高性能元语进行了充分的优化。同时，该镜像集成了Intel® Extension for TensorFlow（简称ITEX），它是一种异构、高性能的深度学习扩展插件，基于TensorFlow的PluggableDevice接口。旨在将Intel CPU或GPU设备引入TensorFlow开源社区，以加速AI工作负载，能够充分利用Intel硬件内部的计算能力。特别的，在第四代英特尔至强可扩展处理器Sapphire Rapids、阿里云规格为g8i的新一代实例中可以利用AMX（Advanced Matrix Extensions）加速能力，可极大提升深度学习训练和推理的性能。
- PyTorch是一个开源的机器学习框架，具有高度灵活性和可扩展性，被广泛用于深度学习任务，如图像分类、目标检测、自然语言处理和生成对抗网络等。同时，该镜像集成了Intel® Extension for PyTorch（简称IPEX），为Intel硬件提供额外的性能提升，针对即时模式和图模式都提供了优化，但是与即时模式相比，PyTorch中的图模式通常会通过操作融合等优化技术获得更好的性能。IPEX通过更全面的图优化进一步增强了它们，并优化利用了在Intel CPU上的AVX-512向量神经网络指令和AMX（Advanced Matrix Extensions），以及在Intel离散GPU上的Intel Xe矩阵扩展（XMX）AI引擎。特别的，在第四代英特尔至强可扩展处理器Sapphire Rapids，阿里云规格为g8i的新一代实例上极大提升在深度学习训练和推理的性能。

系统要求

仅提供基础OS为Anolis OS 23的x86镜像，在阿里云第八代规格为g8i的实例上获取更好的性能体验

在阿里云g8i规格实例使用AMX加速神经网络

1. 如果模型中已经使用BFloat16或者Int8量化，则能够自动启用AMX进行计算加速。
2. 若用户模型未进行对应量化，使用Float32进行训练和推理，可以使用以下方法快速启用AMX加速。

   a. [TensorFlow & PyTorch] 使用oneDNN提供的环境变量

   通过设置环境变量`ONEDNN_DEFAULT_FPMATH_MODE=bf16`来使用AMX, 该方法由oneDNN数据库提供功能支持，不需要用户修改代码，自动识别矩阵乘法，使用AMX的BF16指令集对其进行计算加速，使用非常便捷。

   该方法会对全局中支持的算子进行加速，在输入和输出仍为Float32的情况下，使用BFloat16进行计算。该方法相较于直接量化模型的方法，不能够充分的利用AMX性能，但是胜在便捷，适合进行模型性能的快速评估和上线。

   b. [PyTorch] IPEX的自动混精功能AMP(Automatic mixed precision)

   只需要使用ipex.optimize函数即可实现PyTorch模型的快速BFloat16量化以及其他Intel硬件优化。以下代码以Resnet50的推理为例。

```
Plain Text
```

```
import torch
import torchvision.models as models

model = models.resnet50(weights='ResNet50_Weights.DEFAULT')
model.eval()
data = torch.rand(1, 3, 224, 224)

#################### code changes ####################
import intel_extension_for_pytorch as ipex
model = ipex.optimize(model, dtype=torch.bfloat16)
######################################################

with torch.no_grad(), torch.cpu.amp.autocast():
  model(data)
```

   c. [TensorFlow] ITEX的高级自动混精功能(Advanced Auto Mixed Precision)

   ITEX除了完全兼容TensorFlow原生框架中的Keras混合精度API外，还提供了性能更好的高级自动混精功能(Advanced Auto Mixed Precision)，可以通过环境变量和PythonAPI两种方式实现Advanced AMP。

```
export ITEX_AUTO_MIXED_PRECISION=1
export ITEX_AUTO_MIXED_PRECISION_DATA_TYPE="BFLOAT16"
```

```
import intel_extension_for_tensorflow as itex

auto_mixed_precision_options = itex.AutoMixedPrecisionOptions()
auto_mixed_precision_options.data_type = itex.BFLOAT16

graph_options = itex.GraphOptions(auto_mixed_precision_options=auto_mixed_precision_options)
graph_options.auto_mixed_precision = itex.ON

config = itex.ConfigProto(graph_options=graph_options)
itex.set_config(config)
```

d. [Pytorch & TensorFlow] 使用INC（Intel® Neural Compressor）进行混合精度优化

Intel® Neural Compressor是Intel推出的一个开源深度学习模型压缩Python库，旨在在主流框架（如TensorFlow、PyTorch、ONNX Runtime和MXNet）以及Intel扩展（如Intel® Extension for TensorFlow和Intel® Extension for PyTorch）上提供流行的模型压缩技术，例如量化、修剪（稀疏性）、蒸馏和神经架构搜索。

以下以TensorFlow为例，展示如何使用INC进行模型量化。

```
from neural_compressor import mix_precision
from neural_compressor.config import MixedPrecisionConfig

conf = MixedPrecisionConfig() # default precision is bf16
converted_model = mix_precision.fit(model, conf=conf)
converted_model.save('./path/to/save/')
```

注意事项

- 推荐阿里云第八代实例规格为g8i的实例，以获得最佳的性能和用户体验。
- 请确保您已充分阅读 TensorFlow、PyTorch 以及相应优化框架 ITEX、IPEX 的使用许可，并严格按照许可协议使用