天数智芯

智铠加速卡标准测试参考指南

版本：V4.0.0-MR

日期：2024.4.12

适用产品：智铠 50| 智铠 100

1 声明

1.1 版权声明

1.2 免责声明

天数智芯可以随时对本文档或本文档中描述的产品进行改进和/或更改。本文档包括与天数智芯产品有关的信息，作为说明典型应用的一种方式, 因此，不一定提供足以进行生产设计的完整信息。对于本文档中内容的准确性或完整性，天数智芯不做任何陈述或保证。

1.3 联系方式

地址：上海闵行区陈行公路 2168 号 3 幢

电话：021-68886607

网址：www.iluvatar.com

Contents

2 智铠加速卡标准测试参考指南

2.1 修订记录

- COREX01-MR400-RF01-00：2024/4/12

  文档本次发布内容与 V3.2.0-MR 文档相比 (COREX01-MR320-RF01-00)，有以下更新：

  - 更新本次发布测试所用的软件版本
  - 更新本次发布测试的步骤
  - 本次发布测试不包含以下测试：
    * Bert-Base-ner 推理模型测试
    * Faiss 推理模型测试
    * 48 小时压力测试
    * svGPU 功能测试

2.2 安装与卸载软件栈

下表是支持在智铠加速卡上运行的软件说明：

编号	名称	版本	用途
1	Linux	Ubuntu 20.04	软件测试平台搭建
2	CUDA	10.2.89	NVIDIA 软件工具包
3	SDK	4.0.0	天数智芯驱动、软件栈
4	SDK Test Samples	4.0.0	天数智芯测试工具
5	PyTorch	2.1.1	深度学习训练框架
6	ONNXRuntime	1.16.0	深度学习训练框架
7	IGIE	0.9.1	深度学习推理框架
8	IxRT	0.9.0	深度学习推理引擎
9	ResNet50	4.0.0	分类模型
10	VGG16	4.0.0	分类模型
11	BERT-Base	4.0.0	NLP 模型
12	BERT-Large	4.0.0	NLP 模型
13	YOLOv3	4.0.0	检测模型
14	YOLOv5m	4.0.0	检测模型
15	YOLOv5s	4.0.0	检测模型

编号	名称	版本	用途
16	YOLOv7	4.0.0	检测模型
17	YOLOX	4.0.0	检测模型
18	Conformer	4.0.0	语言识别
19	Transformer	4.0.0	NLP 模型

请联系您的应用工程师安装与卸载天数智芯驱动和软件栈。

2.3 智铠加速卡测试指南

2.3.1 兼容性

加速卡软硬件信息、实施功耗、计算及显存占用率可以被系统和虚拟机（Docker）识别。BMC 中也可以查询到加速卡的软硬件信息。

2.3.1.1 测试步骤

1. 登录到服务器后打开终端，输入 **ixsmi**。
2. 登录服务器并启用虚拟机，如 Docker，在虚拟机中输入 **ixsmi**。
3. 打开 BMC 管理页面，进入 **系统管理** - **PCIe 设备**查询 GPU 信息。

2.3.1.2 测试结果

- 系统及虚拟机内均可查询到加速卡信息。
- BMC 的 **PCIe 设备**页面中可看到 GPU 信息。

2.3.2 通信带宽

带宽测试工具：bandwidthTest 工具提供 GPU memcpy 带宽以及 PCIe 中的 memcpy 带宽数据，可以测试 host 和 device，以及 device 间的传输带宽。

P2P 性能测试工具：p2pBandwidthTest 工具提供基于 DMA 的 P2P 传输性能测试。

2.3.2.1 前提条件

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。

2.3.2.2 测试步骤

天数智芯测试工具的安装将随天数智算软件栈同时安装。关于如何使用测试工具，请您参考《测试工具使用指南》。

注意：重要提示

BIOS 中的设置 **IOMMU** (x86 架构) / **SMMU** (ARM 架构) 会造成 P2P 测试带宽下降。为了提升 P2P 测试性能，在测试之前，请参考下图关闭主机 BIOS 中的 **IOMMU** / **SMMU** 设置。

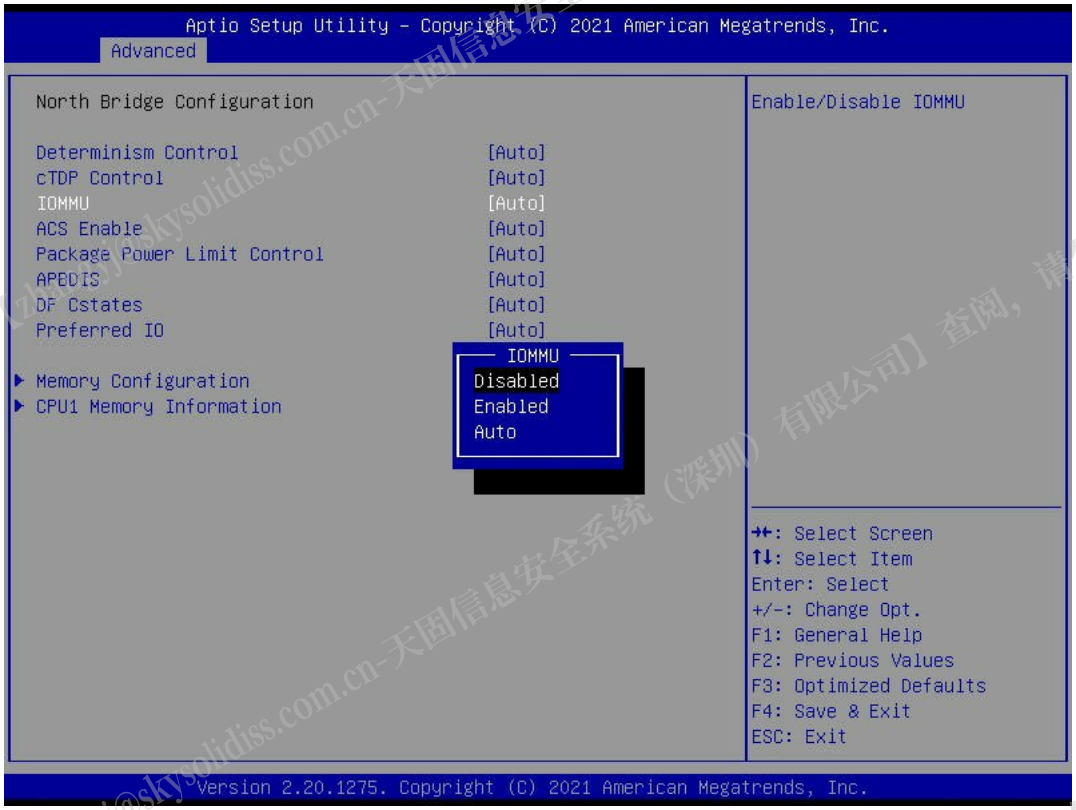

Figure 1: 关闭 IOMMU 设置

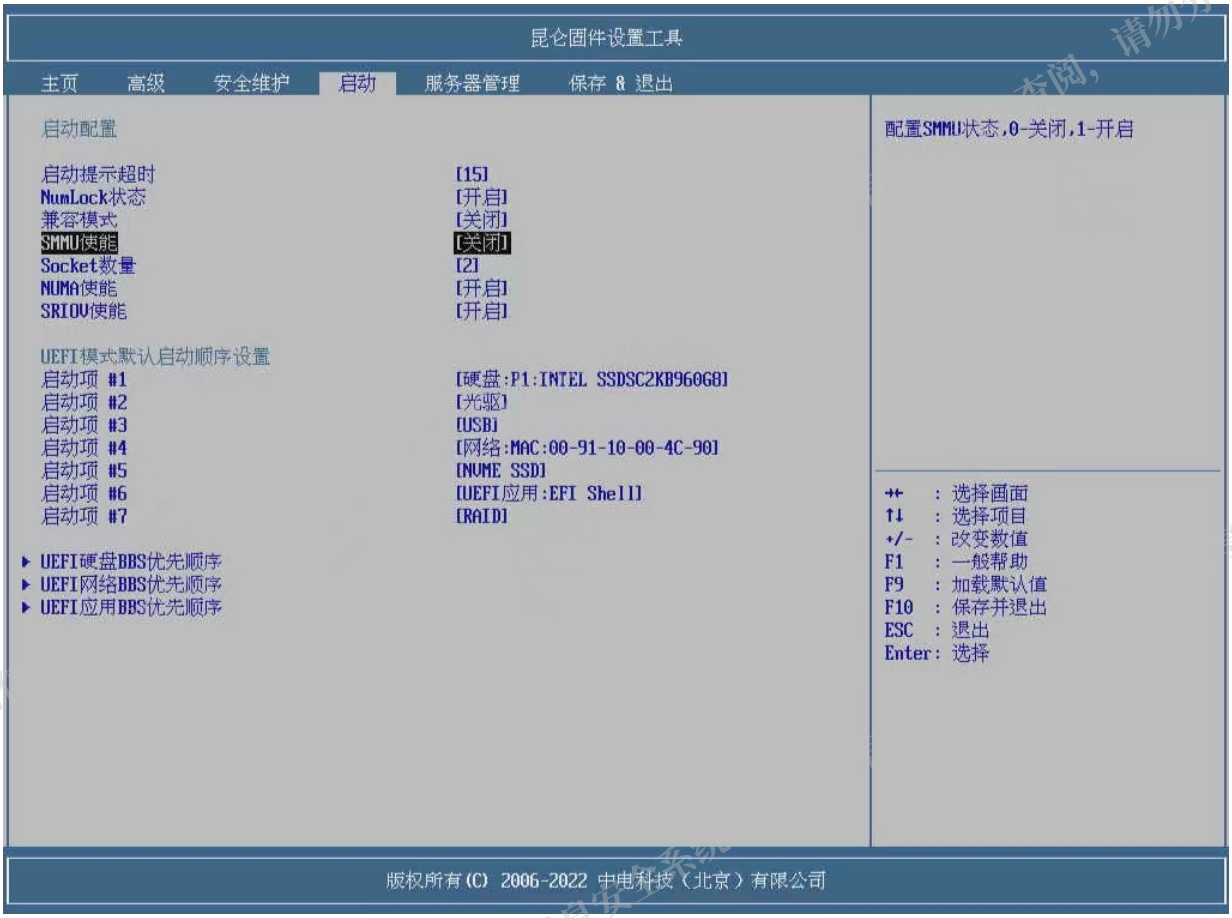

Figure 2: 关闭 SMMU 设置

Note

- 运行 bandwidthTest 工具可以测试 device to host、host to device 及同一个 device 内存传输带宽。
- 运行 p2pBandwidthTest 工具可以测试不同 device 之间数据传输带宽。

2.3.3 算力测试

矩阵算力测试工具：gemm_perf 工具执行用户指定数据类型、循环次数和矩阵大小的 GEMM 运算。

2.3.3.1 前提条件

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。

2.3.3.2 测试步骤

使用 gemm_perf 工具进行矩阵算力测试：

```
$ cd /usr/local/corex-{v.r.m}/extras/test_demo/
$ ./gemm_perf --d 0 --i <Specified device> # 测试 FP32 算力
$ ./gemm_perf --d 1 --l 1000 --i <Specified device> # 测试 FP16 算力
$ ./gemm_perf --d 2 --l 1000 --i <Specified device> # 测试 INT8 算力
```

Tip

您可以参考《测试工具使用指南》获取更多相关内容。

2.3.4 功耗测试

2.3.4.1 前提条件

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。

2.3.4.2 测试步骤

使用 ixsmi 工具进行功耗测试：

```
$ cd /usr/local/corex-{v.r.m}/extras/test_demo/
$ ./gemm_perf --d 0 --l 1000 --i <Specified device> # 脚本可以测试 FP32 算力，同时在另一个终端窗
↪ 口运行 ixsmi dmon -i 0 命令监控记录 GPU0 功耗信息（平均功耗）
$ ./gemm_perf --d 1 --l 100000 --i <Specified device> # 脚本可以测试 FP16 算力，同时在另一个终端
↪ 窗口运行 ixsmi dmon -i 0 命令监控记录 GPU0 功耗信息（平均功耗）
$ ./gemm_perf --d 2 --l 100000 --i <Specified device> # 脚本可以测试 INT8 算力，同时在另一个终端
↪ 窗口运行 ixsmi dmon -i 0 命令监控记录 GPU0 功耗信息（平均功耗）
```

2.3.5 视频解码性能

利用 decTestApp 和 FFmpeg 工具，对 H264 1080p 格式的视频进行解码测试。

2.3.5.1 使用 decTestApp 解码

前提条件

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。
- 请向您的应用工程师获取视频流及测试工具：
  - H264 视频流：nature1920x1080.h264
  - decTestApp：来自天数智芯测试工具
  - 测试脚本：test_dec.py

测试步骤

1. 将 nature1920x1080.h264、decTestApp 与 test_dec.py 放在同一个目录下 (当前版本，decTestApp 默认位于 /usr/local/corex-{v.r.m}/extras/test_demo/ 目录下)。
2. 执行解码：

```
$ cd <path to test_dec.py>
$ python3 test_dec.py --instance_num 100 --no_out --input nature1920x1080.h264
```

Note

- 最后输出的 FPS 表示 100 路解码的平均 FPS，如果 FPS□30，表示解码测试通过。
- 执行 test_dec.py 脚本前输入 **export IX_DRV_FPS=1**，将直接打印每一路的 FPS 值作为参考。

2.3.5.2 使用 FFmpeg 解码

前提条件

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。
- 请向您的应用工程师获取视频流及测试工具：
  - H264 视频流：nature1920x1080.h264
  - FFmpeg：FFmpeg （需要根据 readme 编译安装 FFmpeg）
  - 测试脚本：test_ffmpeg.py

测试步骤

1. 将 nature1920x1080.h264 与 test_ffmpeg.py 放在同一个目录下。
2. 执行解码：

```
$ cd <path to test_ffmpeg.py>
$ export LD_LIBRARY_PATH=/usr/local/ffmpeg/lib:$LD_LIBRARY_PATH
$ export PATH=/usr/local/ffmpeg/bin:$PATH
$ python3 test_ffmpeg.py --data nature1920x1080.h264 --instance_num 30
```

2.3.6 模型推理

通过 IGIE 推理框架和 IxRT 推理引擎对以下模型进行推理测试，从精度与性能两方面考量推理结果。

2.3.6.1 部署环境

- 请确保您已正常安装天数智芯测试工具。
- 请确保您已正常安装天数智芯驱动和软件栈。
- 请向您的应用工程师获取数据集压缩包、推理框架、推理引擎安装包等资源，并按照以下步骤部署环境：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples
$ tar -xvf corex-inference-data-{v.r.m}.tar # 向应用工程师获取 corex-inference-data 压缩包
↪ 并解压
$ bash quick_build_environment.sh
```

2.3.6.2 ResNet50 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/resnet/
$ bash init.sh
$ bash infer_resnet50_int8_accuracy_igie.sh --bs 1
$ bash infer_resnet50_int8_accuracy_ixrt.sh --bs 1
$ bash infer_resnet50_int8_performance_igie.sh --bs 1
$ bash infer_resnet50_int8_performance_ixrt.sh --bs 1
$ bash infer_resnet50_int8_accuracy_igie.sh --bs 32
$ bash infer_resnet50_int8_accuracy_ixrt.sh --bs 32
$ bash infer_resnet50_int8_performance_igie.sh --bs 32
$ bash infer_resnet50_int8_performance_ixrt.sh --bs 32
$ bash infer_resnet50_int8_accuracy_igie.sh --bs 64
$ bash infer_resnet50_int8_accuracy_ixrt.sh --bs 64
$ bash infer_resnet50_int8_performance_igie.sh --bs 64
$ bash infer_resnet50_int8_performance_ixrt.sh --bs 64
$ bash infer_resnet50_fp16_accuracy_igie.sh --bs 1
$ bash infer_resnet50_fp16_accuracy_ixrt.sh --bs 1
$ bash infer_resnet50_fp16_performance_igie.sh --bs 1
$ bash infer_resnet50_fp16_performance_ixrt.sh --bs 1
$ bash infer_resnet50_fp16_accuracy_igie.sh --bs 32
$ bash infer_resnet50_fp16_accuracy_ixrt.sh --bs 32
$ bash infer_resnet50_fp16_performance_igie.sh --bs 32
$ bash infer_resnet50_fp16_performance_ixrt.sh --bs 32
$ bash infer_resnet50_fp16_accuracy_igie.sh --bs 64
$ bash infer_resnet50_fp16_accuracy_ixrt.sh --bs 64
$ bash infer_resnet50_fp16_performance_igie.sh --bs 64
$ bash infer_resnet50_fp16_performance_ixrt.sh --bs 64
```

2.3.6.3 VGG16 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/vgg/
$ bash init.sh
$ bash infer_vgg16_int8_accuracy_igie.sh --bs 1
$ bash infer_vgg16_int8_performance_igie.sh --bs 1
$ bash infer_vgg16_int8_accuracy_igie.sh --bs 32
$ bash infer_vgg16_int8_performance_igie.sh --bs 32
$ bash infer_vgg16_int8_accuracy_igie.sh --bs 64
$ bash infer_vgg16_int8_performance_igie.sh --bs 64
$ bash infer_vgg16_int8_accuracy_ixrt.sh --bs 1
$ bash infer_vgg16_int8_performance_ixrt.sh --bs 1
$ bash infer_vgg16_int8_accuracy_ixrt.sh --bs 32
$ bash infer_vgg16_int8_performance_ixrt.sh --bs 32
$ bash infer_vgg16_int8_accuracy_ixrt.sh --bs 64
$ bash infer_vgg16_int8_performance_ixrt.sh --bs 64
```

2.3.6.4 YOLOv3 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/yolov3/
$ bash init.sh
$ bash infer_yolov3_int8_accuracy_igie.sh --bs 1
$ bash infer_yolov3_int8_performance_igie.sh --bs 1
$ bash infer_yolov3_int8_accuracy_igie.sh --bs 32
$ bash infer_yolov3_int8_performance_igie.sh --bs 32
$ bash infer_yolov3_int8_accuracy_igie.sh --bs 64
$ bash infer_yolov3_int8_performance_igie.sh --bs 64
$ bash infer_yolov3_int8_accuracy_ixrt.sh --bs 1
$ bash infer_yolov3_int8_performance_ixrt.sh --bs 1
$ bash infer_yolov3_int8_accuracy_ixrt.sh --bs 32
$ bash infer_yolov3_int8_performance_ixrt.sh --bs 32
$ bash infer_yolov3_int8_accuracy_ixrt.sh --bs 64
$ bash infer_yolov3_int8_performance_ixrt.sh --bs 64
```

2.3.6.5 YOLOv5m 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/yolov5m/
$ bash init.sh
$ bash infer_yolov5m_int8_accuracy_igie.sh --bs 1
```

```
$ bash infer_yolov5m_int8_performance_igie.sh --bs 1
$ bash infer_yolov5m_int8_accuracy_igie.sh --bs 32
$ bash infer_yolov5m_int8_performance_igie.sh --bs 32
$ bash infer_yolov5m_int8_accuracy_igie.sh --bs 64
$ bash infer_yolov5m_int8_performance_igie.sh --bs 64
$ bash infer_yolov5m_int8_accuracy_ixrt.sh --bs 1
$ bash infer_yolov5m_int8_performance_ixrt.sh --bs 1
$ bash infer_yolov5m_int8_accuracy_ixrt.sh --bs 32
$ bash infer_yolov5m_int8_performance_ixrt.sh --bs 32
$ bash infer_yolov5m_int8_accuracy_ixrt.sh --bs 64
$ bash infer_yolov5m_int8_performance_ixrt.sh --bs 64
```

2.3.6.6 YOLOv5s 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/yolov5s/
$ bash init.sh
$ bash infer_yolov5s_fp16_accuracy_igie.sh --bs 1
$ bash infer_yolov5s_fp16_performance_igie.sh --bs 1
$ bash infer_yolov5s_fp16_accuracy_igie.sh --bs 32
$ bash infer_yolov5s_fp16_performance_igie.sh --bs 32
$ bash infer_yolov5s_fp16_accuracy_igie.sh --bs 64
$ bash infer_yolov5s_fp16_performance_igie.sh --bs 64
$ bash infer_yolov5s_fp16_accuracy_ixrt.sh --bs 1
$ bash infer_yolov5s_fp16_performance_ixrt.sh --bs 1
$ bash infer_yolov5s_fp16_accuracy_ixrt.sh --bs 32
$ bash infer_yolov5s_fp16_performance_ixrt.sh --bs 32
$ bash infer_yolov5s_fp16_accuracy_ixrt.sh --bs 64
$ bash infer_yolov5s_fp16_performance_ixrt.sh --bs 64
```

2.3.6.7 YOLOv7 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/yolov7/
$ bash init.sh
$ bash infer_yolov7_int8_accuracy_igie.sh --bs 1
$ bash infer_yolov7_int8_performance_igie.sh --bs 1
$ bash infer_yolov7_int8_accuracy_igie.sh --bs 32
$ bash infer_yolov7_int8_performance_igie.sh --bs 32
```

```
$ bash infer_yolov7_int8_accuracy_igie.sh --bs 64
$ bash infer_yolov7_int8_performance_igie.sh --bs 64
$ bash infer_yolov7_int8_accuracy_ixrt.sh --bs 1
$ bash infer_yolov7_int8_performance_ixrt.sh --bs 1
$ bash infer_yolov7_int8_accuracy_ixrt.sh --bs 32
$ bash infer_yolov7_int8_performance_ixrt.sh --bs 32
$ bash infer_yolov7_int8_accuracy_ixrt.sh --bs 64
$ bash infer_yolov7_int8_performance_ixrt.sh --bs 64
```

2.3.6.8 YOLOX 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/yolox/
$ bash init.sh
$ bash infer_yolox_int8_accuracy_igie.sh --bs 1
$ bash infer_yolox_int8_performance_igie.sh --bs 1
$ bash infer_yolox_int8_accuracy_igie.sh --bs 32
$ bash infer_yolox_int8_performance_igie.sh --bs 32
$ bash infer_yolox_int8_accuracy_igie.sh --bs 64
$ bash infer_yolox_int8_performance_igie.sh --bs 64
$ bash infer_yolox_int8_accuracy_ixrt.sh --bs 1
$ bash infer_yolox_int8_performance_ixrt.sh --bs 1
$ bash infer_yolox_int8_accuracy_ixrt.sh --bs 32
$ bash infer_yolox_int8_performance_ixrt.sh --bs 32
$ bash infer_yolox_int8_accuracy_ixrt.sh --bs 64
$ bash infer_yolox_int8_performance_ixrt.sh --bs 64
```

2.3.6.9 Bert-Base-squad 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/bert/
$ bash init.sh
$ bash infer_bert_base_squad_fp16_accuracy_igie.sh --bs 1
$ bash infer_bert_base_squad_fp16_performance_igie.sh --bs 1
$ bash infer_bert_base_squad_fp16_accuracy_igie.sh --bs 32
$ bash infer_bert_base_squad_fp16_performance_igie.sh --bs 32
$ bash infer_bert_base_squad_fp16_accuracy_igie.sh --bs 64
$ bash infer_bert_base_squad_fp16_performance_igie.sh --bs 64
$ bash infer_bert_base_squad_fp16_accuracy_ixrt.sh --bs 1
```

```
$ bash infer_bert_base_squad_fp16_performance_ixrt.sh --bs 1
$ bash infer_bert_base_squad_fp16_accuracy_ixrt.sh --bs 32
$ bash infer_bert_base_squad_fp16_performance_ixrt.sh --bs 32
$ bash infer_bert_base_squad_fp16_accuracy_ixrt.sh --bs 64
$ bash infer_bert_base_squad_fp16_performance_ixrt.sh --bs 64
```

2.3.6.10 Bert-Large-squad 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/bert/
$ bash init.sh
$ bash infer_bert_large_squad_int8_accuracy_igie.sh --bs 1
$ bash infer_bert_large_squad_int8_performance_igie.sh --bs 1
$ bash infer_bert_large_squad_int8_accuracy_ixrt.sh --bs 1
$ bash infer_bert_large_squad_int8_performance_ixrt.sh --bs 1
$ bash infer_bert_large_squad_int8_accuracy_igie.sh --bs 32
$ bash infer_bert_large_squad_int8_performance_igie.sh --bs 32
$ bash infer_bert_large_squad_int8_accuracy_ixrt.sh --bs 32
$ bash infer_bert_large_squad_int8_performance_ixrt.sh --bs 32
$ bash infer_bert_large_squad_int8_accuracy_igie.sh --bs 64
$ bash infer_bert_large_squad_int8_performance_igie.sh --bs 64
$ bash infer_bert_large_squad_int8_accuracy_ixrt.sh --bs 64
$ bash infer_bert_large_squad_int8_performance_ixrt.sh --bs 64
$ bash infer_bert_large_squad_fp16_accurary_igie.sh --bs 1
$ bash infer_bert_large_squad_fp16_performance_igie.sh --bs 1
$ bash infer_bert_large_squad_fp16_accuracy_ixrt.sh --bs 1
$ bash infer_bert_large_squad_fp16_performance_ixrt.sh --bs 1
$ bash infer_bert_large_squad_fp16_accurary_igie.sh --bs 32
$ bash infer_bert_large_squad_fp16_performance_igie.sh --bs 32
$ bash infer_bert_large_squad_fp16_accuracy_ixrt.sh --bs 32
$ bash infer_bert_large_squad_fp16_performance_ixrt.sh --bs 32
$ bash infer_bert_large_squad_fp16_accurary_igie.sh --bs 64
$ bash infer_bert_large_squad_fp16_performance_igie.sh --bs 64
$ bash infer_bert_large_squad_fp16_accuracy_ixrt.sh --bs 64
$ bash infer_bert_large_squad_fp16_performance_ixrt.sh --bs 64
```

2.3.6.11 Conformer 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/conformer/
$ bash init.sh
$ bash infer_conformer_fp16_accuracy_igie.sh --batch_size 1
$ bash infer_conformer_fp16_performance_igie.sh --batch_size 1
$ bash infer_conformer_fp16_accuracy_igie.sh --batch_size 32
$ bash infer_conformer_fp16_performance_igie.sh --batch_size 32
$ bash infer_conformer_fp16_accuracy_igie.sh --batch_size 64
$ bash infer_conformer_fp16_performance_igie.sh --batch_size 64
$ bash infer_conformer_fp16_accuracy_ixrt.sh --batch_size 1
$ bash infer_conformer_fp16_performance_ixrt.sh --batch_size 1
$ bash infer_conformer_fp16_accuracy_ixrt.sh --batch_size 32
$ bash infer_conformer_fp16_performance_ixrt.sh --batch_size 32
$ bash infer_conformer_fp16_accuracy_ixrt.sh --batch_size 64
$ bash infer_conformer_fp16_performance_ixrt.sh --batch_size 64
```

2.3.6.12 Transformer 推理

测试步骤

进入模型脚本目录并执行：

```
$ cd /root/corex-samples-{v.r.m}_{arch}/samples/inferencesamples/executables/transformer/
$ bash init.sh
$ bash infer_transformer_fp16_accuracy_igie.sh --batch-size 1
$ bash infer_transformer_fp16_performance_igie.sh --max_batch_size 1
$ bash infer_transformer_fp16_accuracy_igie.sh --batch-size 32
$ bash infer_transformer_fp16_performance_igie.sh --max_batch_size 32
$ bash infer_transformer_fp16_accuracy_igie.sh --batch-size 64
$ bash infer_transformer_fp16_performance_igie.sh --max_batch_size 64
$ bash infer_transformer_fp16_accuracy_ixrt.sh --batch-size 1
$ bash infer_transformer_fp16_performance_ixrt.sh --max_batch_size 1
$ bash infer_transformer_fp16_accuracy_ixrt.sh --batch-size 32
$ bash infer_transformer_fp16_performance_ixrt.sh --max_batch_size 32
$ bash infer_transformer_fp16_accuracy_ixrt.sh --batch-size 64
$ bash infer_transformer_fp16_performance_ixrt.sh --max_batch_size 64
```

3 商标声明

- 天数智芯、天数智芯 logo、Iluvatar CoreX 等商标、标识、组合商标为上海天数智芯半导体有限公司之注册商标或商标，受法律保护。
- 除了天数智芯的注册商标外，本内容中使用的其他产品名称及标志仅用于识别目的，该等名称及标志可能是归属于其各自公司的商标。我们否认对该等名称及标志的所有权利。
- CentOS 标识为 Red Hat 公司的商标。
- Docker 为 Docker 公司在美国和其他国家的商标或注册商标。
- Linux 为 Linus Torvalds 在美国和其它国家的注册商标。
- NVIDIA 和 CUDA 为 NVIDIA 公司在美国和/或其它国家的商标和/或注册商标。
- PyTorch 为 Facebook 公司的商标。
- TensorFlow 为 Google 公司的商标。
- Ubuntu 为 Canonical 公司的注册商标。