**License 加密镜像创建**

**一、配置目录和环境变量**

|  |
| --- |
| Bash # 加密镜像 export OBS\_IMAGE=zcr.chatglm.cn/zhipu/text-generation-inference-privacy:240125-obfuscator-codegen  export OBS\_IMAGE=zcr.chatglm.cn/zhipu/text-generation-inference:240207-ccb  export OBS\_IMAGE=zcr.chatglm.cn/infra/text-generation-inference:240308  # Megatron 模型目录(可选) #export MGT\_MODEL\_DIR=<Megatron模型目录> # TGI 模型目录 export TGI\_MODEL\_DIR=/mnt/dev/zhangxinze/model/chatglm-130b-128k-v0.1-awq-tgi  # 分片数（卡数） export NUM\_SHARD=8  # obfuscator目录  export obfuscator=/mnt/dev/zhangxinze/test # SHARD 模型目录 export SHARD\_MODEL\_DIR=/mnt/dev/zhangxinze/shardModel-chatglm-130b-128k-v0.1  # 加密模型目录 export ENC\_MODEL\_DIR=/mnt/dev/zhangxinze/encModel-chatglm-130b-128k-v0.1  # 当前日期 export DATE=$(date '+%y%m%d') # 镜像TAG export TAG=$(cat /dev/urandom | tr -dc '0-9a-z' | head -c 8) # 加密随机种子 export SEED=$(cat /dev/urandom | tr -dc '0-9' | head -c 20) |

**二、模型格式转换和加密**

1. 将 Megatron 模型转换成 TGI 模型 （根据模型格式决定是否运行，可能跳过）

|  |
| --- |
| Bash docker run --rm --shm-size 32g -v $MGT\_MODEL\_DIR:/mgt\_model -v $TGI\_MODEL\_DIR:/tgi\_model --entrypoint python3 $OBS\_IMAGE server/text\_generation\_server/models/glm2/convert.py --model\_dir /mgt\_model --save\_dir /tgi\_model |

1. 将 TGI 格式分片，转换为 SHARD 格式

|  |
| --- |
| Bash docker run --rm --gpus all --shm-size 32g -v $TGI\_MODEL\_DIR:/tgi\_model -v $SHARD\_MODEL\_DIR:/shard\_model -e QUANT\_GROUP\_SIZE=64 --entrypoint python3 $OBS\_IMAGE server/text\_generation\_server/models/glm2/shard-convert.py --model\_dir /tgi\_model --save\_dir /shard\_model --num\_shard $NUM\_SHARD --quantize glmquantize4gzkv --dtype float16  docker run --rm --gpus all --shm-size 32g -v $TGI\_MODEL\_DIR:/tgi\_model -v $SHARD\_MODEL\_DIR:/shard\_model -v $obfuscator:/obfuscator --entrypoint python3 $OBS\_IMAGE /obfuscator/shard-convert.py --model\_dir /tgi\_model --save\_dir /shard\_model --num\_shard $NUM\_SHARD --quantize glmquantize8gzkv --dtype float16 |

1. 对 SHARD 格式加密，得到加密模型，并得到WEIGHT\_SALT和WEIGHT\_SCALE\_SALT以及codegen.cpp

|  |
| --- |
| Bash docker run --rm --shm-size 32g -v $SHARD\_MODEL\_DIR:/shard\_model -v $ENC\_MODEL\_DIR:/model -v $obfuscator:/obfuscator --entrypoint python3 $OBS\_IMAGE /obfuscator/obfuscate.py --model\_dir /shard\_model --save\_dir /model --seed $SEED > codegen.cpp  docker run --rm --shm-size 32g -v $SHARD\_MODEL\_DIR:/shard\_model -v $ENC\_MODEL\_DIR:/model --entrypoint python3 $OBS\_IMAGE server/text\_generation\_server/models/glm2/obfuscate.py --model\_dir /shard\_model --save\_dir /model --seed $SEED > codegen.cpp |

1. 该命令会产生一个codegen.cpp，命令行中会有"weight\_salt: <WEIGHT\_SALT> weight\_scale\_salt: <WEIGHT\_SCALE\_SALT>"的字样。
2. 将WEIGHT\_SALT和WEIGHT\_SCALE\_SALT加入环境变量

|  |
| --- |
| Bash export WEIGHT\_SALT=2039678894 export WEIGHT\_SCALE\_SALT=44332 |
|  |

1. 进入glm\_privacy分支，将产生的codegen.cpp放入text-generation-inference/server/glm-models/csrc目录。
2. 构建可解密加密模型的镜像，使用当前日期和镜像TAG来标记

|  |
| --- |
| Bash docker build --network=host -t tgi-privacy-temp:$DATE-$TAG --build-arg ALL\_PROXY=socks5://10.1.2.128:1080 --build-arg http\_proxy=http://10.1.2.128:1087 --build-arg https\_proxy=http://10.1.2.128:1087 --build-arg LICENSE\_TAG=$TAG --build-arg WEIGHT\_SALT=$WEIGHT\_SALT --build-arg WEIGHT\_SCALE\_SALT=$WEIGHT\_SCALE\_SALT --build-arg ENABLE\_OBFUSCATE=”true” --build-arg VIRBOX\_PROTECTOR\_IP=10.1.2.180 . |

1. ~~进行.so的加壳等，有待后续自动化改造（已自动化，跳过此过程）~~

|  |
| --- |
| Bash docker run --rm --entrypoint sleep --name tgi-privacy-temp-240311-n203sa94 tgi-privacy-temp:240311-n203sa94 100000  docker run --network=host --privileged --entrypoint sleep --name tgi-privacy-temp-240307-ywhr49tn tgi-privacy-temp:240307-ywhr49tn 100000000  hostnetwork # 有待补充 docker cp tgi-privacy-temp-$DATE-$TAG:/opt/conda/lib/python3.9/site-packages/glm\_privacy\_model ./ docker cp tgi-privacy-temp-$DATE-$TAG:/opt/conda/lib/python3.9/site-packages/glm\_privacy\_kernel ./ docker cp ./usb\_network\_gate\_x64-2.deb tgi-privacy-temp-240305-gw6krqcb:/usr/src/ docker cp ./obfuscator tgi-privacy-temp-240311-n203sa94:/opt/conda/lib/python3.9/site-packages docker export tgi-privacy-temp-$DATE-$TAG | docker import --change 'ENTRYPOINT ["text-generation-launcher"]' --change 'ENV PATH /opt/conda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin' - zcr.chatglm.cn/zhipu/text-generation-inference-privacy:$DATE-$TAG  docker export tgi-privacy-temp-240307-ywhr49tn-hostnetwork | docker import --change 'ENTRYPOINT ["text-generation-launcher"]' --change 'ENV PATH /opt/conda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin' - zcr.chatglm.cn/zhipu/text-generation-inference-privacy:240307-ywhr49tn-hostnetwork  docker stop tgi-privacy-temp-$DATE-$TAG |

1. 记录加密相关信息至[License统计表](https://zhipu-ai.feishu.cn/base/IIeKb4ZFAaWtQTsIa2TcucyZnwe?table=tblH64c5IC99xhoD&view=vewDW5uWK7)

|  |
| --- |
| Bash echo $DATE-$TAG echo zcr.chatglm.cn/zhipu/text-generation-inference-privacy:$DATE-$TAG echo $SEED echo $WEIGHT\_SALT echo $WEIGHT\_SCALE\_SALT |

**二、产生License文件等**

1. 获取环境信息

|  |
| --- |
| Bash #!/usr/bin/perl  use strict; use warnings;  my $board\_name = `cat /sys/class/dmi/id/board\_name | tr -d '\n'`; my $board\_serial = `cat /sys/class/dmi/id/board\_serial | tr -d '\n'`; my $board\_vendor = `cat /sys/class/dmi/id/board\_vendor | tr -d '\n'`; my $product\_name = `cat /sys/class/dmi/id/product\_name | tr -d '\n'`; my $product\_version = `cat /sys/class/dmi/id/product\_version | tr -d '\n'`; my $product\_serial = `cat /sys/class/dmi/id/product\_serial | tr -d '\n'`; my $product\_uuid = `cat /sys/class/dmi/id/product\_uuid | tr -d '\n'`;  my $output = "board\_name:$board\_name,board\_serial:$board\_serial,board\_vendor:$board\_vendor,product\_name:$product\_name,product\_version:$product\_version,product\_serial:$product\_serial,product\_uuid:$product\_uuid\\0";  open(my $fh, '>', 'license\_env.txt'); print $fh "$output"; close $fh; |

注意点：

* 该脚本可暴露给客户，让客户执行。
* 每个推理机器都需要执行一次，收集对应的硬件信息。每次执行的输出是license\_env.txt文件。
* 例如：有50个License授权，那就应该获取50个license\_env.txt文件
* 多个机器，生成的多个license\_env.txt，整理放到一个目录里，以备后续授权使用。

1. 执行license\_sign签发license，参数：镜像tag、有效期（days）
2. 将License文件放入 $ENC\_MODEL\_DIR 中
3. 加密模型目录 $ENC\_MODEL\_DIR 为最终交付模型，将tokenizer等文件拷入

|  |
| --- |
| Bash  cp $TGI\_MODEL\_DIR/\*token\* $ENC\_MODEL\_DIR cp $SHARD\_MODEL\_DIR/\*token\* $ENC\_MODEL\_DIR echo '{"model\_type": "chatglm2privacy"}' > $ENC\_MODEL\_DIR/config.json |

**三、测试镜像、License、模型**

|  |
| --- |
| Bash # 启动服务 docker run -it --rm --name tgi-privacy-test-$DATE-$TAG  \  --gpus=all -e ENABLE\_PREFILL\_LOGPROBS=1 \  -v $ENC\_MODEL\_DIR:/model \  -p 8080:8080 \  zcr.chatglm.cn/zhipu/text-generation-inference-privacy:$DATE-$TAG  \  --model-id /model \  --num-shard=8 \  --port=8080 \  --max-concurrent-requests=1024 \  --max-input-length=2000 \  --max-total-tokens=2048 \  --max-batch-prefill-tokens=2048 \  --trust-remote-code \  --max-waiting-tokens=2 \  --waiting-served-ratio=0.1  # 测试模型输出 # 3代Prompt curl 127.0.0.1:8080/generate\_stream \  -X POST \  -d '{  "inputs": "<|user|>\n你好<|assistant|>",  "parameters": {  "best\_of": 1,  "decoder\_input\_details": true,  "details": true,  "do\_sample": true,  "temperature": 0.95,  "top\_p": 0.7,  "max\_new\_tokens": 64,  "return\_full\_text": false,  "stop": ["<|user|>", "</s>", "<|assistant|>", "<eos>"],  "seed": null  } }' \  -H 'Content-Type: application/json' |

**四、进行交付**

1. 上传镜像

|  |
| --- |
| Bash docker push zcr.chatglm.cn/zhipu/text-generation-inference-privacy:$DATE-$TAG |

1. 上传模型至B/G等团队服务器，并进行记录

rsync -avzrPL ./ [modelDeliver@117.50.172.227::upload/](mailto:modelDeliver@117.50.172.227::upload/)

docker run -it --rm --gpus \"device=0,1,2,3\" --shm-size 32g -p 8080:8080 -v /data/delivery/66b/:/model 27e6c2cbbbc9 --model-id /model/ --num-shard 4 --port 8080 --max-concurrent-requests 409600 --max-input-length 2048 --max-total-tokens 4096 --max-batch-prefill-tokens 2048 --trust-remote-code --max-waiting-tokens 2 --waiting-served-ratio 0.2 --block-size 32 --quantize glmquantize8kv --cuda-memory-fraction 0.98

curl [127.0.0.1:8080/generate](http://127.0.0.1:8083/generate" \t "/Users/zxz/Documents\\x/_blank)\ -X POST \ -H 'Content-Type: application/json' \ -d '{ "inputs": "<|user|>\n下列哪个句子的主语是被省略了的？\n{1.常死不了的孩子，常常最后会当上大官。\n 2.你别太担心，总会有月亮的。\n3.雨一直下着，他提着伞。\n4.走在夜路上，有点害怕。}\n<|assistant|>\n", "parameters": { "do\_sample": true, "top\_p": 0.8, "max\_new\_tokens": 1000, "stop": ["<|endoftext|>","<|user|>","<|observation|>"] , "seed": null } }'

curl [127.0.0.1:8080/generate](http://127.0.0.1:8083/generate" \t "/Users/zxz/Documents\\x/_blank)\ -X POST \ -H 'Content-Type: application/json' \ -d '{ "inputs": "<|user|>\n介绍一下李白}\n<|assistant|>\n", "parameters": { "do\_sample": true, "top\_p": 0.8, "max\_new\_tokens": 1000, "stop": ["<|endoftext|>","<|user|>","<|observation|>"] , "seed": null } }'