Relation Network for Few-shot Learning

——基于多卡的并行加速

刘佳玮，计算机科学与技术学院，20031211496

https://muyuuuu.github.io

2020 年 11 月 20 日

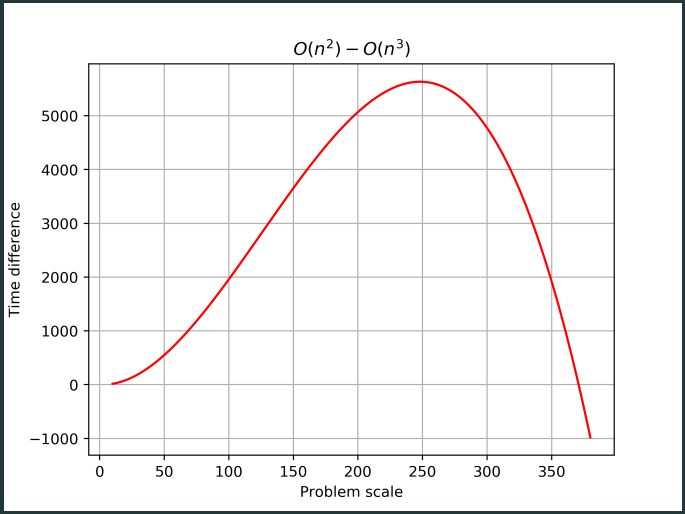
大纲

1. 好的算法与好的设备
2. 发挥设备优势
3. 硬件与软件依赖
4. 模型
5. 并行实现
6. 实验结果

好的算法与好的设备

算法时间复杂度

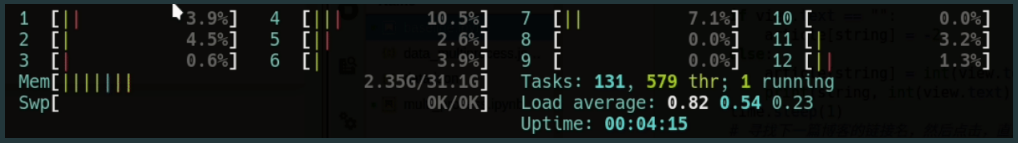
> 同一个问题，一个时间复杂度 $O(n^2)$ 与 $O(n^3)$ 的时间对比，代码开放于：
> https://github.com/muyuuuu/Algorithm/tree/master/Insert_sort



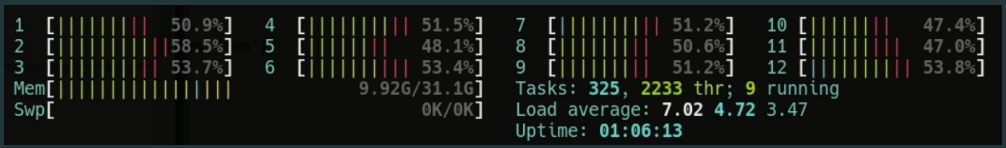
发挥设备优势

发挥设备优势

一个耗时 1153 秒的单进程任务：



使用多进程改进，相同任务耗时 105 秒，且多核利用率较为均衡：



> 任务必须可以并行化。代码地址：
> https://muyuuuu.github.io/2020/03/18/multi-process/

硬件与软件依赖

硬件与软件

硬件部分

CPU 2 个 Intel(R) Xeon(R) Gold 5115 CPU 2.40GHz，10 核心 20 线程

GPU 4 路 Tesla P40，每路显存容量 22GB

内存 128GB

外存 520TB 可用，已用 15TB

软件部分

系统 CentOS Linux release 7.3.1611 (执行), Arch 5.9.6(开发)

python 3.8.2，开发语言

pytorch 1.6.0，模型实现，借助其提供的 API 实现并行

ssh OpenSSH_8.3p1, OpenSSL 1.1.1h：实现远程登录

scp 文件传输

常用命令

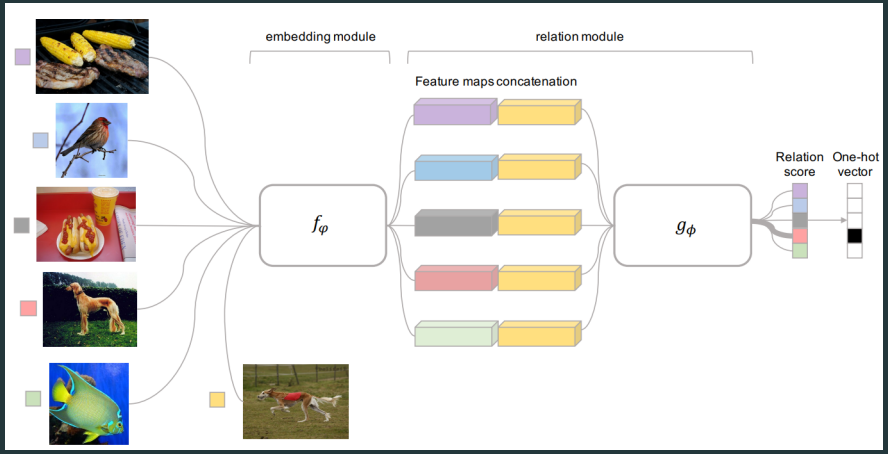
命令行内执行

1. `mv, cd, ls, cp, cat` 等文件操作
2. `nohup python train.py > log` 挂起运行与重定向输出
3. `ps -f|grep python` 查看挂起程序是否执行

模型

模型结构

实现的模型为 Relation Network[1]。数据集为 miniImageNet[2]。



[1] https://ieeexplore.ieee.org/abstract/document/8778601
[2] https://drive.google.com/file/d/0B3Irx3uQNoBMQ1FlNXJsZUdYWEE/view

最终模型结构

> 共计 11,285,569 个参数，约 75MB 左右的参数。

13 组特征提取模块，7 组连接处理模块

```
(conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
(conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
(relu): ReLU(inplace=True)
(0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
```

1 组计算相关性模块

```
(0): Linear(in_features=256, out_features=64, bias=True)
(1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True)
(2): ReLU(inplace=True)
(3): Linear(in_features=64, out_features=1, bias=True)
(4): Sigmoid()
```

并行实现

多角度并行

- 保持实验参数一致。

并行加载数据

1. `DataLoader，num_workers` 指定加载数据进程的数量。
2. `.cuda()` 在指定设备上设置和运行 CUDA 操作，使其支持 CUDA 张量类型；之后便可以使用 GPU 完成计算。

并行训练模型

1. 单机单卡
2. 单机多卡
3. 多机多卡

并行实现

> 模型过大，因此不进行 CPU 实验和 GPU 单卡实验，只对比单机多卡或多机多卡下不同并行方式的加速比。

- `DataParallel`[3]，给定模型，将输入划分给不同的设备。前向计算阶段，每个设备复制一份模型，读取自己的输入并执行；反向传播阶段，每个设备的 loss 汇总到原始模型（指定设备的模型），计算梯度并重新分配下去。
- `DistributedDataParallel`[4]，DDP 使用集群通信来同步梯度和缓冲区的数据。DDP 对模型中每一个可求导的参数申请一个钩子，当对应的参数在反向传播中计算梯度时，钩子会触发信号，DDP 发射信号后来同步不同的进程间的数据[5]，阻塞等待所有参数更新完毕。

[3] https://pytorch.org/docs/stable/generated/torch.nn.DataParallel.html
[4] https://pytorch.org/tutorials/intermediate/ddp_tutorial.html
[5] https://pytorch.org/tutorials/intermediate/dist_tuto.html

DataParallel 与 DistributedDataParallel 对比

DistributedDataParallel

- 多进程实现
- 支持多机
- 支持模型并行，将模型拆分，并放到多个机器
- 通信方式：`Allreduce`

DataParallel

- 单进程多线程实现，由于 GIL 锁的存在，效率低下
- 只能用于单机
- 通信方式：scattering inputs and gathering outputs.

并行部分伪代码

设备检测，即如何监测多卡

```
if device_ids is None:
    device_ids = _get_all_device_indices()

if output_device is None:
    output_device = device_ids[0]
```

DataParallel

```
net = DataParallel(Compare(n_way, k_shot)).cuda()
```

DistributedDataParallel

```
1. initialize the process group
2. net = DDP(Compare(n_way, k_shot).to(rank), device_ids=[rank])
3. destroy_process_group
```

实验结果

DataParallel 单机多卡

▌ nvidia-smi 查看显卡利用率：

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P40           On   | 00000000:3B:00.0 Off |                    0 |
| N/A   55C    P0   157W / 250W |  22825MiB / 22919MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P40           On   | 00000000:86:00.0 Off |                    0 |
| N/A   58C    P0   185W / 250W |  22015MiB / 22919MiB |     91%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla P40           On   | 00000000:AF:00.0 Off |                    0 |
| N/A   37C    P0   138W / 250W |  11736MiB / 22919MiB |     51%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla P40           On   | 00000000:D8:00.0 Off |                    0 |
| N/A   33C    P0   151W / 250W |   6451MiB / 22919MiB |     28%      Default |
+-------------------------------+----------------------+----------------------+
```

程序执行时间：$T_1 = 137172$ 秒，约 2286 分钟，约 1.59 天。

DistributedDataParallel 单机多卡

▌ nvidia-smi 查看显卡利用率：

文件(F)　编辑(E)　视图(V)　书签(B)　设置(S)　帮助(H)

```
Wed Nov 18 21:56:15 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P40           On   | 00000000:3B:00.0 Off |                    0 |
| N/A   63C    P0   146W / 250W |   5891MiB / 22919MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P40           On   | 00000000:86:00.0 Off |                    0 |
| N/A   57C    P0    94W / 250W |  22839MiB / 22919MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla P40           On   | 00000000:AF:00.0 Off |                    0 |
| N/A   52C    P0    53W / 250W |  22449MiB / 22919MiB |     95%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla P40           On   | 00000000:D8:00.0 Off |                    0 |
| N/A   55C    P0    96W / 250W |  22239MiB / 22919MiB |     89%      Default |
+-------------------------------+----------------------+----------------------+
```

程序执行时间：$T_2 = 89856$ 秒，约 1498 分钟，约 1.03 天。

DistributedDataParallel 单机多卡

- 加速比：$\frac{T_1}{T_2} = 1.53$
- 准确率对比：Dataparallel：0.566，DDP：0.582。
- 代码开放于：https://github.com/muyuuuu/Algorithm/tree/master/meta-learning/Metric-based/Relation-Netowrk

感谢聆听！