Effective OzSTAR 中文版

本文档中包含使用OzSTAR进行研发的实用技巧。开始之前，请确保熟悉OzSTAR官方文档和OzSTAR Tutoral的内容。

存储

如File systems and I/O所述，每个用户将获得对两个文件夹的管理权。其中，`/home/<user_name>`中的存储空间十分有限，因此我们建议在其中仅保留必要文件，并将体积较大的文件和目录转移至空间更大的`/fred/<project_id>`中。这些大文件和目录包括但不限于：

- 系统临时文件夹
- conda 虚拟环境及环境中安装的库
- 体积较大的项目文件（如保存的深度学习模型及可视化文件存档）

其中，系统临时文件夹的转移可以在`/home/<user_name>/.bashrc`中进行配置：

```
export TMPDIR="/fred/<project_id>/tmp"
export TEST_TMPDIR="/fred/<project_id>/test_tmp/"
mkdir -p $TMPDIR
```

conda 虚拟环境及库的转移可在`/home/<user_name>/.condarc`中进行配置。

```
envs_dirs:
  - /fred/<project_id>/anaconda/envs/

pkgs_dirs:
  - /fred/<project_id>/anaconda/pkgs/
```

修改配置文件后重新登录账号，即可安全删除对应的旧的文件夹。

包的使用

本章将会介绍Anaconda、GIT和CUDA包的使用和注意事项。普通用户无sudo权限，无法自行安装包。常用的包已经在系统中预装，可通过`module load`命令进行加载。如有额外需要，可发邮件至hpc-support@swin.edu.au联系管理员进行安装。

Conda Environment

加载conda模块，并激活某个虚拟环境：

```
module load anaconda3/5.1.0
source activate <vir_env_name>
```

目前系统中的最新版本为`anaconda3/5.1.0`，用户可选择加载自己需要的版本。为方便起见，建议将此条命令加入`.bashrc`，以使其在登录后自动加载。Conda虚拟环境的基本配置和库安装与其他平台和系统一致，可参考Anaconda官方教程。

GIT

推荐使用GIT来管理项目文件。首先加载GIT模块：

```
module load git/2.18.0
```

可以在项目文件夹中使用credential来管理项目仓库的密码，方便同步：

```
cd <path_to_the_project_dir>
git config credential.helper store
```

类似的，用户可以按需要加载不同版本的GIT，并将上述命令加入`.bashrc`来省去每次登陆后重新配置的麻烦。

CUDA

如需使用GPU，需加载CUDA库，在深度学习项目中，CuDNN库也是必要的。

加载CUDA：

```
module load cuda/10.1.243
```

同时加载CUDA和CuDNN：

```
module load cudnn/7.6.5-cuda-10.1.243
```

一些软件需要CUDA中的附加库才能运行，但这些附加库没有被自动链接，这意味着我们有时需要将其手动添加到环境变量中，如CUPIT:

```
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:"/apps/skylake/software/CUDA/10.1.243/extras/CUPTI/lib64"
export
LD_INCLUDE_PATH=$LD_INCLUDE_PATH:"/apps/skylake/software/CUDA/10.1.243/extras/CUPTI/include"
```

推荐将上述命令加入`.bashrc`。

一些软件使用内置的路径名来推测CUDA及CuDNN的位置，但这些位置可能并不适用于OzSTAR。一个解决方法是在项目目录下创建指向CUDA库的symbol link：

```
ln -s /apps/skylake/software/CUDA/10.1.243/bin/ bin
ln -s /apps/skylake/software/CUDA/10.1.243/nvvm/ nvvm
```

代码调试

OzSTAR的登录节点（farnarkle1或者2）为交互式节点（interavtive node），该节点硬件配置与普通的计算机相同，可用于代码调试。可使用如下命令在这两个节点之间切换：

```
ssh f1
ssh f2
```

OzSTAR的登录节点算例较小，且为公用，仅应用于基本操作和测试。实际运行规模较大、时间较长的项目时，请参考"提交任务"章节将任务提交至Job Queues。

一些程序编辑器提供了ssh-remote功能，可以在服务器端达到与本地类似的开发和调试体验，详情可参考：

- VSCode remote-ssh
- PyCharm Professional：该软件可以student或staff账号免费获取，具体使用方法可参考：
  https://www.jetbrains.com/help/pycharm/remote-debugging-with-product.html

提交任务

任务script的编写和提交及相关例程可参考Job Queues官方文档。除官方文档中所述外，以下几个技巧可以提高用户调试和程序运行的效率：

将程序输出保存为文件

首先，我们可以将任务运行时的输出保存至本地的文件，具体做法是在job script文件中加入：

```
#SBATCH --output <log_file_path>
#SBATCH --error <log_file_path>
```

job script中参数名的含义与Python程序中的有所不同。实际测试中，由`print`产生的输出将保存至`--output`指定的文件中，由`logging`产生的输出将被保存至`--error`指定的文件中。两个文件可以是同一个，程序在任务节点上产生的输出将被实时保存，可以在登录节点上使用命令：

```
tail -f <log_file_path>
```

来重现这些输出。

使用临时高速存储区

当对磁盘文件的读写成为了程序运行的瓶颈，我们可以考虑在程序运行开始之前，将文件拷贝至IO效率更高的任务节点。我们可以在job script中实现这个功能：

```
#SBATCH --tmp 16G

srun -N $SLURM_NNODES -n $SLURM_NNODES cp -a <src_data_dir> "$JOBFS"

srun python <program_path> --data_dir="$JOBFS"
```

首先，我们通过`--tmp`来申请一块足够大的磁盘空间，这块空间可以通过`$JOBFS`来访问。然后，我们将登录节点的数据集文件拷贝至这块空间，需要注意的是如果将文件拷贝至`$JOBFS`中某个文件夹中，需预先创建这个文件夹，否则拷贝命令无法成功。最后，当运行程序时，需提供指定的路径，来让程序在任务节点上找到这些文件。

可视化

计算性能可视化

OzSTAR Job Monitor可以可视化任务的排队及运行情况，包括运行时长、资源占用等等数据。

深度学习可视化

TensorBoard可以方便地可视化对于深度学习任务中的数据。TensorBoard基于网页显示，因此，在远程运行时，需要将服务器的网页数据回传至本地。在此，我们需要用到SSH的端口转发（port forwarding）功能。使用方法如下：

```
ssh -L 16006:127.0.0.1:6006 <user_name>@ozstar.swin.edu.au
tensorboard --logdir <log_dir>
```

在登录时，指定将服务器的`6006`端口数据，转发至本地（`127.0.0.1`）的`16006`端口。这样，在启动Tensorboard后，即使用本地的浏览器通过http://127.0.0.1:16006/查看服务器端的可视化结果。

使用6006的原因是Tensorboard默认将数据发送至该端口。如果这个端口已经被占用，可以通过设置使其发送至其他端口：

```
ssh -L 16006:127.0.0.1:6007 <user_name>@ozstar.swin.edu.au
tensorboard --logdir <log_dir> --port 6007
```

在此特别提醒，转发端口的占用难以预知，为避免数据的泄露及减少对其他人造成的麻烦，请务必在无需查看TensorBoard时，及时关闭！