Deep Learning(CS280) Tutorial:
SIST AI-Cluster

2022/09/14

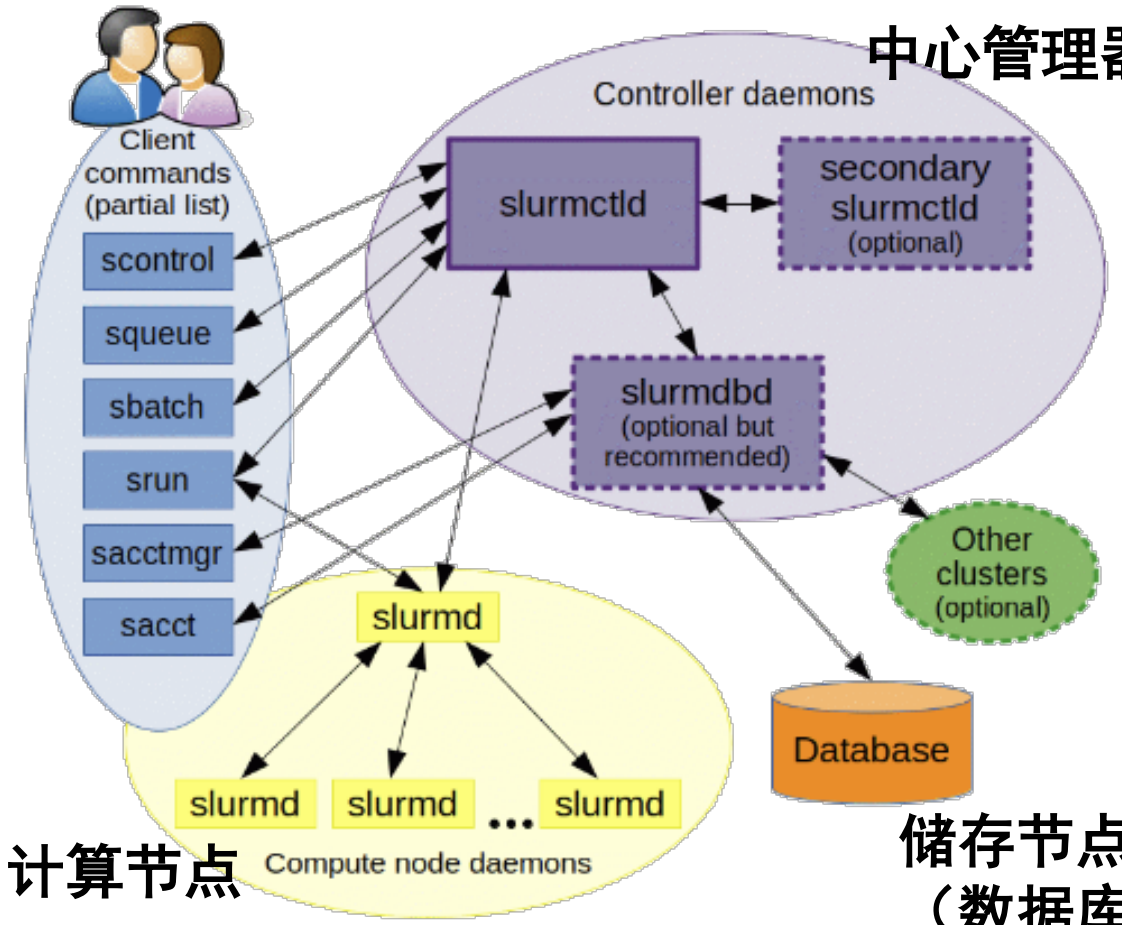
蒋苏一

Overview

- 集群架构概览
- 登陆
- 远程调试代码

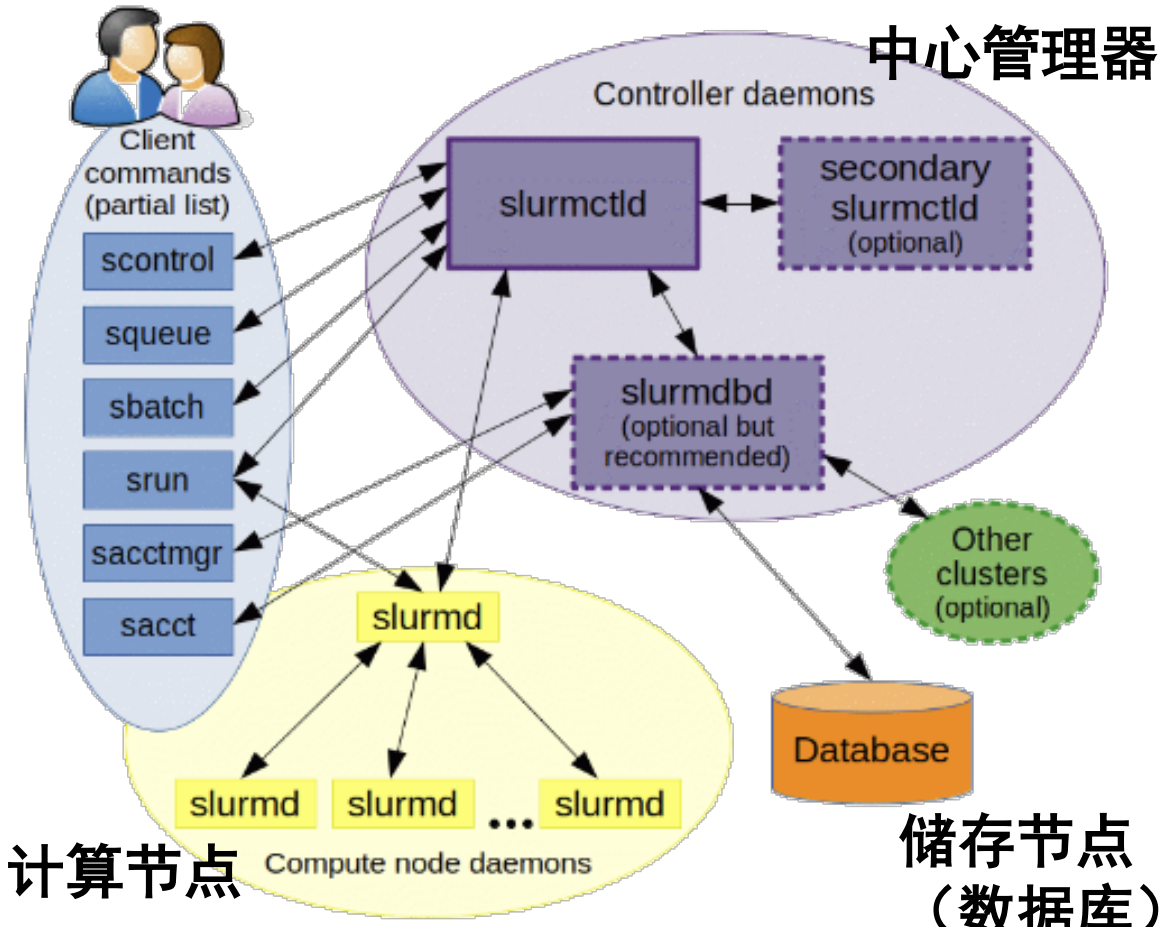
集群架构概览

- 每节点独立运行，仅节点上运行的任务会产生IO开销，并且不直接依赖于存储，系统的稳定性更强。
- 用户通过系统账号登录，配合NIS可以方便实现基于用户和用户组的目录权限管理。
- 支持任务的资源调度和排队系统，能更好地分配资源。
- 支持历史作业信息导出，方便收集和统计数据。



集群架构概览

- 用户不再具有管理员权限，因此apt相关命令和直接修改系统参数不再支持
- 通用软件，统一安装并更新到所有节点中（如gcc、glibc、CUDA等），个性化软件，只采用源码编译的方式安装（如zsh）



计算资源 – 硬件

- 1个管理节点（曙光R620-G20）、2个登录节点（戴尔R730，可访问外网）、
- 共66个计算节点（不可访问外网）

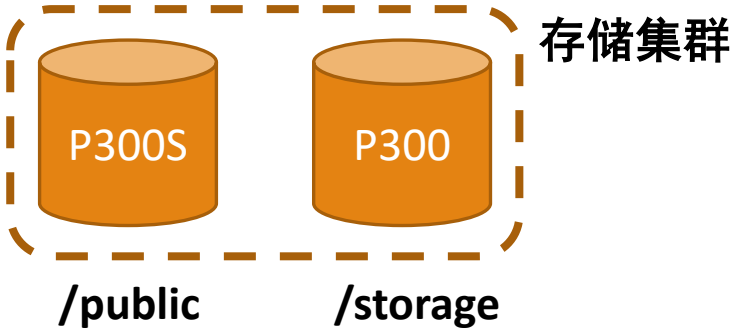
登录节点
10.15.89.191
10.15.89.192
端口：22112

- 1套曙光P300S存储系统（6台存储节点）和1套曙光P300存储系统（2台管理/索引节点+13台存储节点）

- **可通过登陆节点下载，配置环境，但不要在登录节点跑计算程序**

存储集群

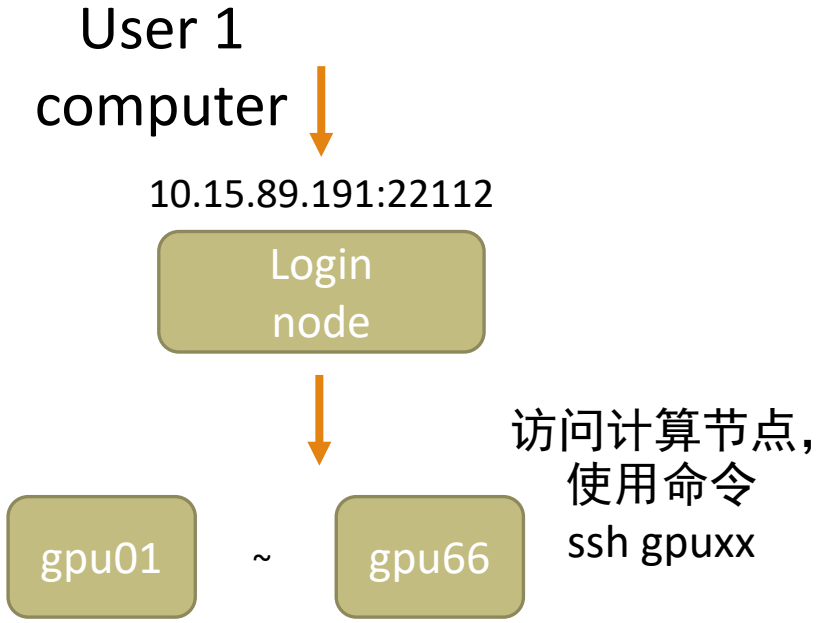
- P300S:
  - 用于存放软件、计算数据、基础环境等
  - 用户的home文件夹，1T
- P300
  - 存放历史数据，2T
- 注意事项
  - **P300S的调度策略更好，像模型的checkpoint，training data这些东西可以放到P300S中去，可以使用soft link将p300的文件夹挂载到home目录的文件夹内：**
  - `ln -s /<somefolder> <the_target_mount_point>`



权限与网络架构

- **校级、重点保障科研项目，如protein_folding队列**
- **院级主要科研项目，对应critical队列**
- **一般日常科研工作，对应normal队列**
- **课程授课、本科生毕业设计等。**

队列名	说明
protein_folding	ai_hgx01~05，蛋白质折叠项目专用，无特殊作业限制
debug(default)	ai_gpu01~35，默认队列，用于用户Debug任务，用户限制提交并运行1个作业，每个作业最大运行时长为2小时，每个作业限制4个CPU核心和2张显卡
critical	ai_gpu01~60，每个用户限制运行20个作业，限制提交40个作业，每个作业最大运行时长为15天，每个用户限制使用16张显卡
normal	ai_gpu42~66，每个用户限制运行10个作业，限制提交20个作业，每个作业最大运行时长为10天，每个用户限制使用8张显卡



注意：在计算节点没有正在运行的作业存在时，无法直接通过ssh登录计算节点

登录

Linux/Unix/Mac用户 ssh,scp,sftp等

可以使用终端中的命令行工具登录。下列语句指出了该节点的IP地址、用户名、SSH端口和密钥。

```
$ ssh YOUR_USERNAME@TARGET_IP:PORT -i <Path of private key>
```

推荐配置ssh-config-file，可参考
https://linuxize.com/post/using-the-ssh-config-file/

```
Host 10.15.89.125
  HostName 10.15.89.125
  User root
  Port 30071
  IdentityFile C:\Users\user\.ssh\id_rsa
```

登录节点
10.15.89.191
10.15.89.192
端口：22112

ssh <user>@10.15.89.191 -p 22112

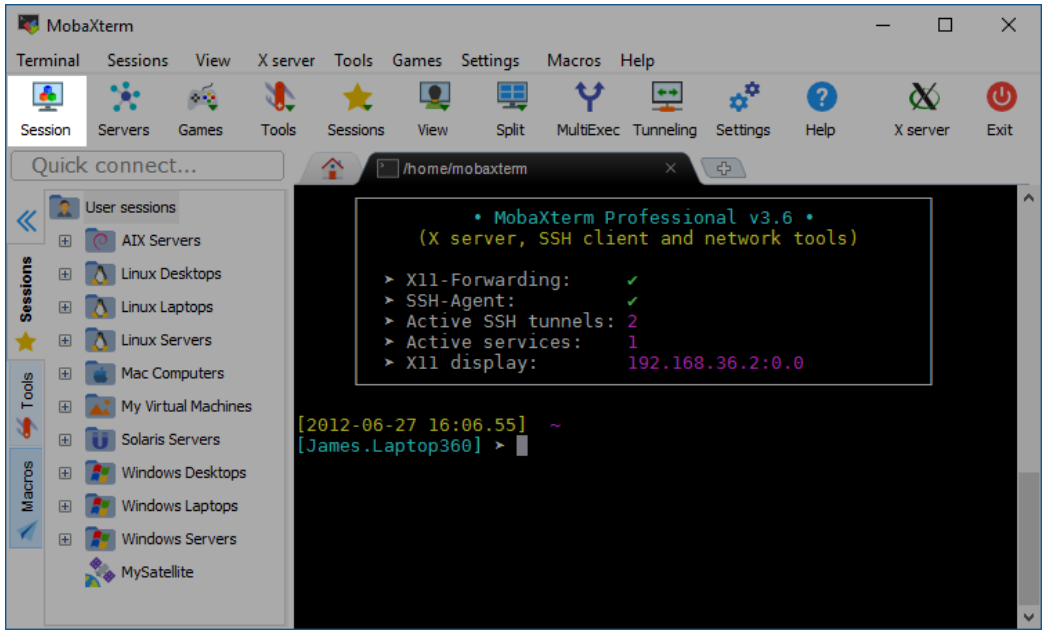
- **集群ip为校园网内网，登录时本机IP需要是学校IP.**
- 除此以外还有 termius, putty, xshell 和 vscode 等可用于 ssh 连接

登录

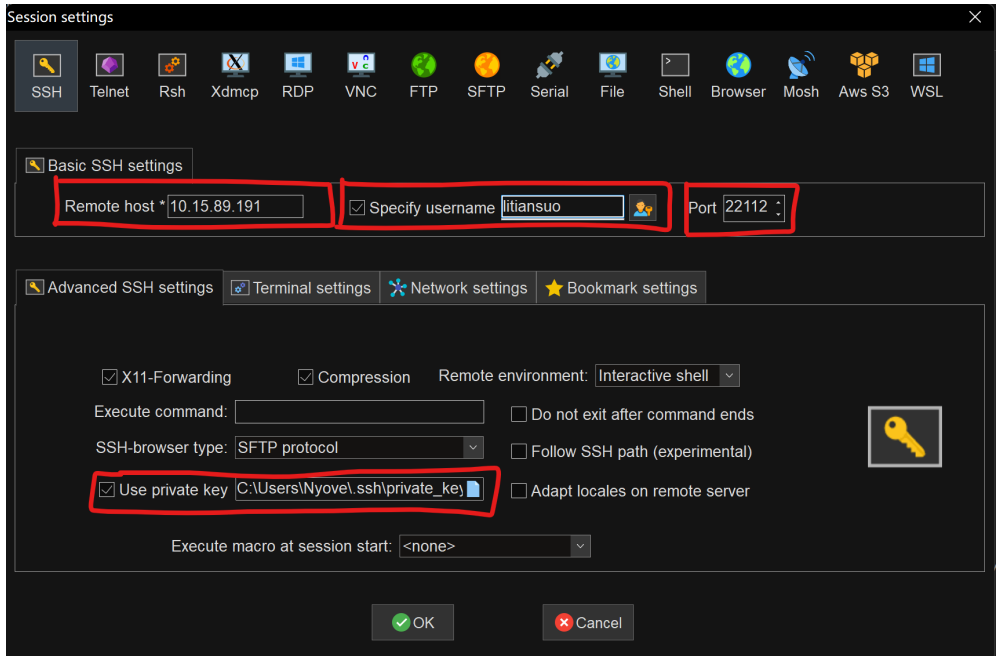
Windows用户

推荐使用 mobaxterm 免费客户端，可至 mobaxterm 下载

启动 mobaxterm ，并在菜单栏的左上角找到 sesion



之后依次填入 远程ip 用户 端口 私钥文件路径



除此以外还有 termius, putty, xshell 和 vscode 等可用于 ssh 连接

Slurm 作业机制

- **scontrol show partition， 查看AllowAccounts来了解所有权限**
- **sinfo, sinfo -N, 查看整体资源以及节点所开放的权限**
- **sinfo -O Nodehost,Gres:.30,GresUsed:.45**

```
$ sinfo
   PARTITION       AVAIL  TIMELIMIT  NODES  STATE NODELIST
   protein_folding    up   infinite      5   idle gpu[01-05]
   debug*             up   infinite      2   idle gpu[06-07]
```

```
$ sinfo -N
   NODELIST   NODES       PARTITION STATE
   gpu01          1 protein_folding idle
   gpu02          1 protein_folding idle
   gpu03          1 protein_folding idle
   gpu04          1 protein_folding idle
   gpu05          1 protein_folding idle
   gpu06          1          debug* idle
   gpu07          1          debug* idle
```

```
$ sinfo -O Nodehost,Gres:.30,GresUsed:.45
   HOSTNAMES           GRES                GRES_USED
   gpu01               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu02               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu03               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu04               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu05               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu06               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
   gpu07               gpu:NVIDIAA40:8     gpu:NVIDIAA40:0(IDX:
```

drain(节点故障)，alloc(节点在用)，idle(节点可用)，down(节点下线)，mix(节点部分占用，但仍有剩余资源)

Slurm 作业机制

- **squeue，查看目前正在排队或运行的作业(历史作业记录可以用sacct查看)**
- **squeue -l 展示更细节的信息**

```
$ squeue
   JOBID PARTITION   NAME     USER ST       TIME  NODES NODELIST(REASON)
   389     debug     bash    wentm PD       0:00      1 (QOSMaxGRESPerJob)
   388     debug     bash    wentm  R       0:14      1 gpu06
```

```
$ squeue -l
Thu Dec  9 14:07:17 2021
 JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
   389     debug     bash    wentm  PENDING       0:00   3:00:00      1 (QOSMaxGRESPerJob)
   388     debug     bash    wentm  RUNNING       1:02   3:00:00      1 gpu06
```

作业状态包括R(正在运行)，PD(正在排队)，CG(即将完成)，CD(已完成)。

Slurm 作业机制

- **sbatch，提交作业脚本，脚本后缀.slurm**

$ sbatch run.slurm

```
#!/bin/bash
#SBATCH -J test
#SBATCH -p normal
#SBATCH --cpus-per-task=4
#SBATCH --mail-type=all
#SBATCH --mail-user=YOU@MAIL.COM
#SBATCH -N 1
#SBATCH -t 1-00:00:00
#SBATCH --gres=gpu:<N>   #SBATCH --gres=gpu:TeslaM4024GB:<N>
#SBATCH --output=%j.out
#SBATCH --error=%j.err

hostname
sleep 600
```

https://slurm.schedmd.com/

Slurm	含义
-n [count]	总进程数
–ntasks-per-node=[count]	每台节点上的进程数
-p [partition]	作业队列
–job-name=[name]	作业名
–output=[file_name]	标准输出文件
–error=[file_name]	标准错误文件
–time=[dd-hh:mm:ss]	作业最大运行时长
–exclusive	独占节点
–gres=gpu:N	任务每节点需要N块GPU
–mail-type=[type]	通知类型，可选 all, fail, end，分别对应全通知、故障通知、结束通知
–mail-user=[mail_address]	通知邮箱
–nodelist=[nodes]	偏好的作业节点
–exclude=[nodes]	避免的作业节点
–depend=[state:job_id]	作业依赖
–array=[array_spec]	序列作业

Slurm 作业机制

- **srun&salloc，交互式作业，参数与sbatch一致**
- **srun -N 1 --cpus-per-task=2 -p debug hostname**
- **salloc -N 1 --cpus-per-task=4 -p critical --gres=gpu:1**

  ssh gpuxx

```
$ nvidia-smi
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |                               |                      |               MIG M. |
   |===============================+======================+======================|
   |   0  NVIDIA A40          On   | 00000000:89:00.0 Off |                    0 |
   |  0%   26C    P8    21W / 300W |      0MiB / 45634MiB |      2%      Default |
   |                               |                      |                  N/A |
   +-------------------------------+----------------------+----------------------+

   +-----------------------------------------------------------------------------+
   | Processes:                                                                  |
   |  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
   |        ID   ID                                                   Usage      |
   |=============================================================================|
   |  No running processes found                                                 |
   +-----------------------------------------------------------------------------+
```

资源在使用率低的情况下默认保留一小时，默认情况下，计算程序最长运行24小时

Slurm 作业机制

- **scontrol，用于控制排队或运行的作业**
- **sacct, 用于查看历史作业信息**

scontrol

Slurm	功能
scontrol show job JOB_ID	查看排队或正在运行的作业的信息
scontrol hold JOB_ID	暂停JOB_ID
scontrol release JOB_ID	恢复JOB_ID
scontrol update dependency=JOB_ID	添加作业依赖性，以便仅在JOB_ID完成后才开始作业
scontrol –help	查看所有选项

sacct

Slurm	功能
sacct -l	查看详细的帐户作业信息
sacct –states=R	查看具有特定状态的作业的账号作业信息
sacct -S YYYY-MM-DD	在指定时间后选择处于任意状态的作业
sacct –format="LAYOUT"	使用给定的LAYOUT自定义sacct输出
sacct –help	查看所有选项

Slurm 作业机制

注意

- **默认情况下，用户不允许登录到计算节点，仅当某节点有该用户任务运行时，用户可以登录。但ssh连接会随着任务的中止而断开。**
- **为保证资源及时释放，请勿在计算节点使用tmux或后台命令，如果需要保持登录，请在登录节点上使用tmux并提交任务。**
- **为了更合理分配资源，采用了cgroups限制，用户仅可使用自己申明的资源，因此提交GPU作业时，必须申明所需要使用的GPU卡数。**

计算资源--软件

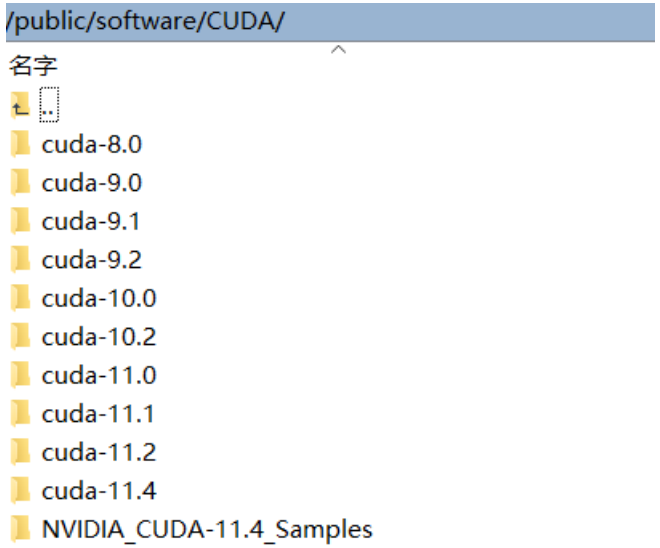
- 集群已有环境(GPU 计算)
  - Nvidia Driver(不支持用户自定义)
  - CUDA Version: 8.0, gcc/g++ Version 4.8.0
  - shell: bash (如需zsh则自行编译安装，可参照 https://gist.github.com/mgbckr/b8dc6d7d228e25325b6dfaa1c4018e78)
- Python环境
  - 自带 python 2.7
  - 推荐自行安装anaconda或miniconda管理虚拟环境，
- 安装环境时…
  - 在登录节点安装
  - 自行编译安装
  - 禁止sudo

计算资源--软件

- 提供常用软件的多个版本
  - 如CUDA, gcc/g++
- 通用软件统一安装在/public/software
  - 引用对应的软件环境，可以参考/public/software/bashrc_example

软件	描述	可用版本
anaconda3	它提供了以跨平台的方式构建、分发、安装、更新和管理软件的实用程序。Conda使管理多个数据环境变得容易，这些环境可以单独维护和运行，互不干扰。	4.10.3
cmake	CMake是个一个开源的跨平台自动化建构系统，用来管理软件建置的程序，并不依赖于某特定编译器，并可支持多层目录、多个应用程序与多个库。 它用配置文件控制建构过程的方式和Unix的make相似，只是CMake的配置文件取名为CMakeLists.txt。	3.22.0
gcc	GNU编译器套装，指一套编程语言编译器，以GPL及LGPL许可证所发行的自由软件，也是GNU计划的关键部分，也是GNU工具链的主要组成部分之一。GCC也常被认为是跨平台编译器的事实标准	10.2.0, 7.5
Open MPI	Open MPI 能够结合整个高性能计算社区的专业知识、技术和资源，以建立现有的最佳MPI库。Open MPI为系统和软件供应商、应用开发商和计算机科学研究人员提供了优势。	1.6.5
automake	GNU Automake是一种编程工具，可以产生供make程序使用的Makefile，用来编译程序。automake所产生的Makefile符合GNU编程标准。 automake是由Perl语言所写的，必须和GNU autoconf一并使用。	1.15, 1.16.5
icc	Intel® C++ Compiler 是一个基于标准的、跨架构的编译器	
gmp	GNU 多精度算术库	4.3.2, 6.2.1
mpc	GNU MPC是一个具有精确四舍五入的复杂浮点库。	0.8.1, 1.2.1
CUDA	CUDA（或称计算统一设备架构）是一个并行计算平台和应用编程接口（API），允许软件使用某些类型的图形处理单元（GPU）进行通用处理–这种方法称为GPU通用计算（GPGPU）。CUDA是一个软件层，可以直接访问GPU的虚拟指令集和并行计算元素，用于执行计算内核。	8.0~11.4
make	GNU Make是一个控制从程序的源文件生成可执行文件和其他非源文件的工具。	4.3
mpfr	MPFR库是一个用于多精度浮点计算的C库，具有正确的舍入功能。	2.4.2, 4.1.0
singularity	singularity是一个容器平台，或者说是一个容器软件。singularity允许您创建和运行容器，在容器内安装调试软件，以可移植、可重现的方式打包软件。	3.5.2

计算资源--软件



~/.bashrc

```
#######CUDA 11.4########
export PATH=/public/software/CUDA/cuda-11.4/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/public/software/CUDA/cuda-11.4/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

/public/software/bashrc_example

计算资源--GPU

- GPU 资源的监控
  - nvidia-smi （第三方的 glances, gpustat）
  - 集群dashboard 网页监控 http://10.15.89.177:8899/gpu
- 指定GPU的使用
  - CUDA_VISIBLE_DEVICES=“gpu_id” e.g. CUDA_VISIBLE_DEVICES=0,1 python ..
  - 自动分配的gpu以0,1,2..顺序排列，与其在节点上的顺序无关
  - 一个自己定义的函数：set_gpu x
  - 如果不指定，默认用0~N号卡
- 结束程序
  - pkill -f <进程名>
  - kill -9 <pid>
  - top 监控工具找到进程

AI Cluster上运行Juypter notebook

使用场景：

- 在gpu_09 的端口32658上运行jupter notebook/jupyter-lab，需要在本地浏览器打开jupter窗口

Step1: 在login节点(191)上开启正向代理

- 命令格式：`ssh –N –f -L 10.15.89.191:32658:0.0.0.0:32658 <user>@ai_gpu09`
  `<port>` 为任意一个未使用的端口号

Step2: 在gpu09上运行 jupyter lab

- ‘juypter-lab --ip=0.0.0.0 --port=32658’

Step3: 在本地浏览器访问：’ 10.15.89.191:32658’

计算资源 – 远程调试代码

Visual Studio Code

- 安装remote-ssh 插件
- 添加远程服务器信息
- 连接到服务器

Pycharm远程调试

- https://cloud.tencent.com/developer/news/221060

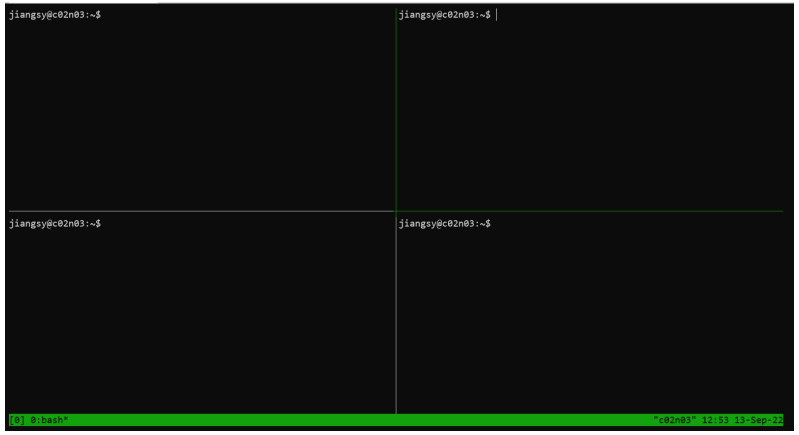
计算资源-远程连接

- 后台运行程序
  - tumx后台运行程序, 关掉当前ssh session的窗口也能保持后台程序的运行, 可以使用鼠标 相关快捷键见教程 https://www.hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/
- 本地与远程文件共享
  - Use sshfs(Unix):

```
sshfs -p port root@10.15.89.41:<target-dir> <local_mount_point> -o reconnection -o compression=no -o follow-symlinks -o sync_read -o sshfs_sync
```

  - SFTP Net Drive 2017 (Windows include windows sub linux)
  - Pycharm remote Deploy
  - **大文件传输使用 scp 命令**

对一个终端窗口任意分割



计算资源-注意事项

- 终止程序后要记得检查相关资源是否已经释放
- 禁止在登录节点跑计算程序
- 禁止恶意占卡

Q&A

有问题欢迎发邮件或者发在piazza上

jiangsy@shanghaitech.edu.cn