Инструкция пользователя GPU-сервера АТП и лаборатории инноватики ФПМИ

Ивченко Олег, Чернецкий Аркадий, 2020-23 г.

Характеристики сервера

- Диски

Точка монтирования	Размер	Тип (HDD / SSD)
/	450 Gb	HDD
/home	2 Tb	HDD
/data	3 Tb	HDD
/data_fast (используется для docker)	440 Gb	SSD
/opt	10 Gb	HDD
/usr/local	50 Gb	HDD

- RAM: 256 Gb,
- CPU: Intel Xeon Gold 6136 v4, 24 ядра,
- GPU: Nvidia GeForce RTX 2080, **8 видеокарт**, видеопамять на каждой 11 Gb.

Установленное ПО

- CUDA 12.2,
- Python 3.10.
- JuPyter с ядром Python 3.10.
- Библиотеки:
  - стандартные: numpy, scipy, pandas, matplotlib,
  - для нейронных сетей: tensorflow, keras, pytorch. Для работы с ними требуется активировать окружение Anaconda - выполните conda-activate. Для работы с Jupyter перед запуском так же активируйте окружение Anaconda

Названия библиотек	Версии
CUDA	10.1.243
CudNN	7.6.4
Tensorflow	2.13
Keras	2.13
Pytorch	1.5.1
miniConda	23.7.2

Доступ на сервер

Для доступа нужно **заполнить** **форму**. Через некоторое время после заполнения вам на указанный email придёт логин-пароль.

Доступ для Unix-систем

Доступ на сервер происходит по ssh: `ssh LOGIN@lorien.atp-fivt.org`.

Для упрощения процедуры доступа можно сделать одно из двух:

1. Добавить в файл /etc/hosts строку:
   `93.175.29.112 lorien.atp-fivt.org lorien`
2. В ~/.ssh/config добавить строки:

```
host lorien
    port 22
    hostname 93.175.29.112
    user <MY_USER>
```

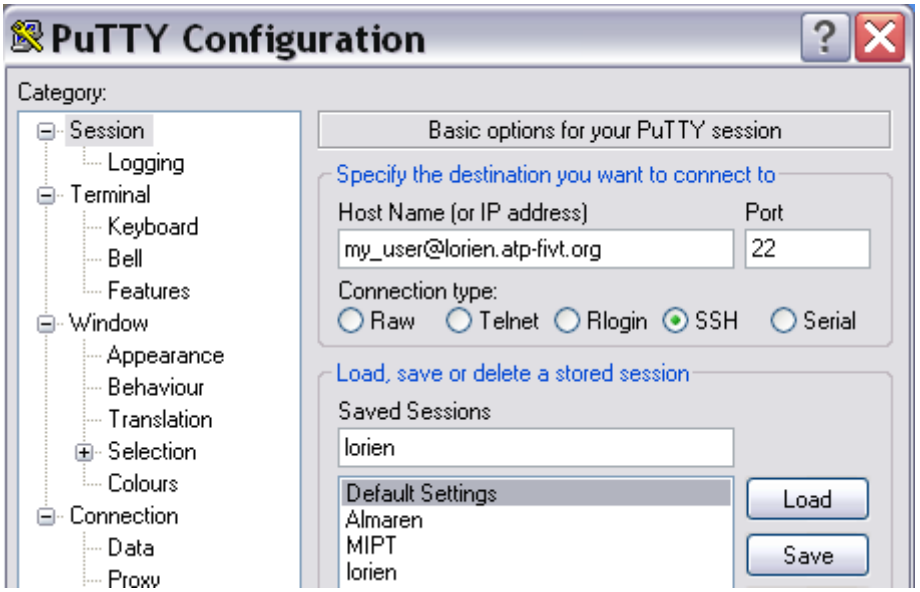
Доступ для Windows-систем

Для доступа из Windows можно использовать Git bash, при установке которой автоматически ставится оболочка MinGW. Внутри MinGW доступ работает так же, как для Unix (см. выше). Если Git-оболочка работает медленно, обратите внимание на:

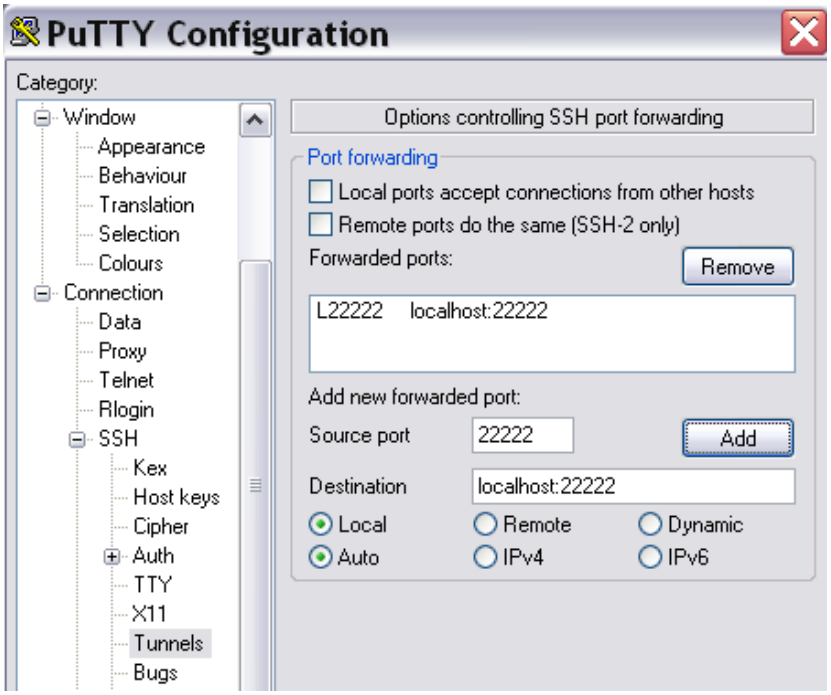
- issue в репозитории Git,
- вопрос на Stackoverflow.

Альтернативный вариант - программа Putty.

1) На вкладке “Session” вводим реквизиты доступа.



2) На вкладке “SSH->Tunnels” настраиваем проброс портов для Jupyter-ноутбуков.

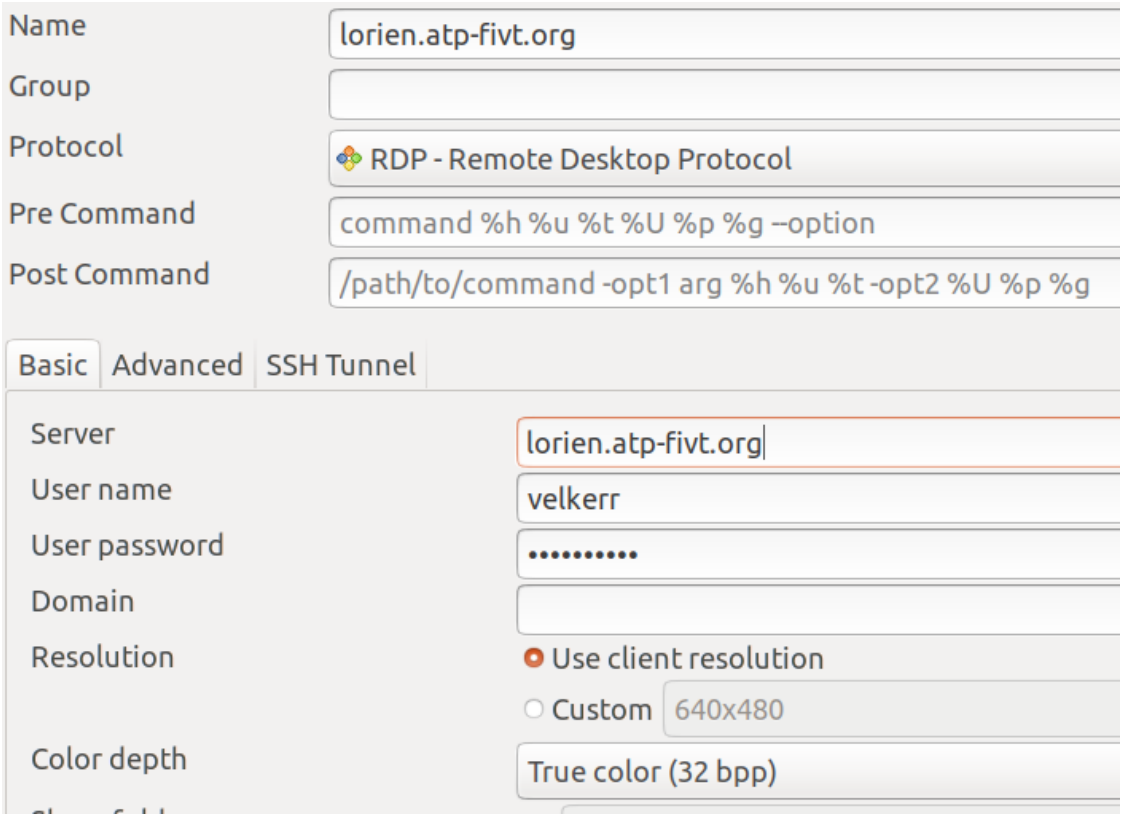


3) Возвращаемся в “Session” и сохраняем настройки (Save). В дальнейшем достаточно будет только открыть Putty, выбрать сохранённую сессию (нажать “Load”) и подключиться (нажать “Open”).

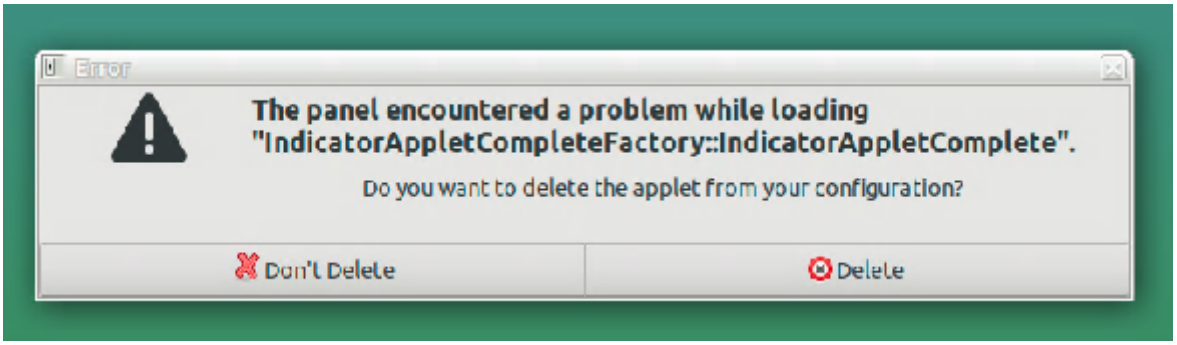
Подключение по RDP (Remmina)

Вход на сервер

Для подключения по RDP рассмотрим инструмент Remmina. Введите такие параметры как указано на скриншоте (бОльшая часть параметров по умолчанию).



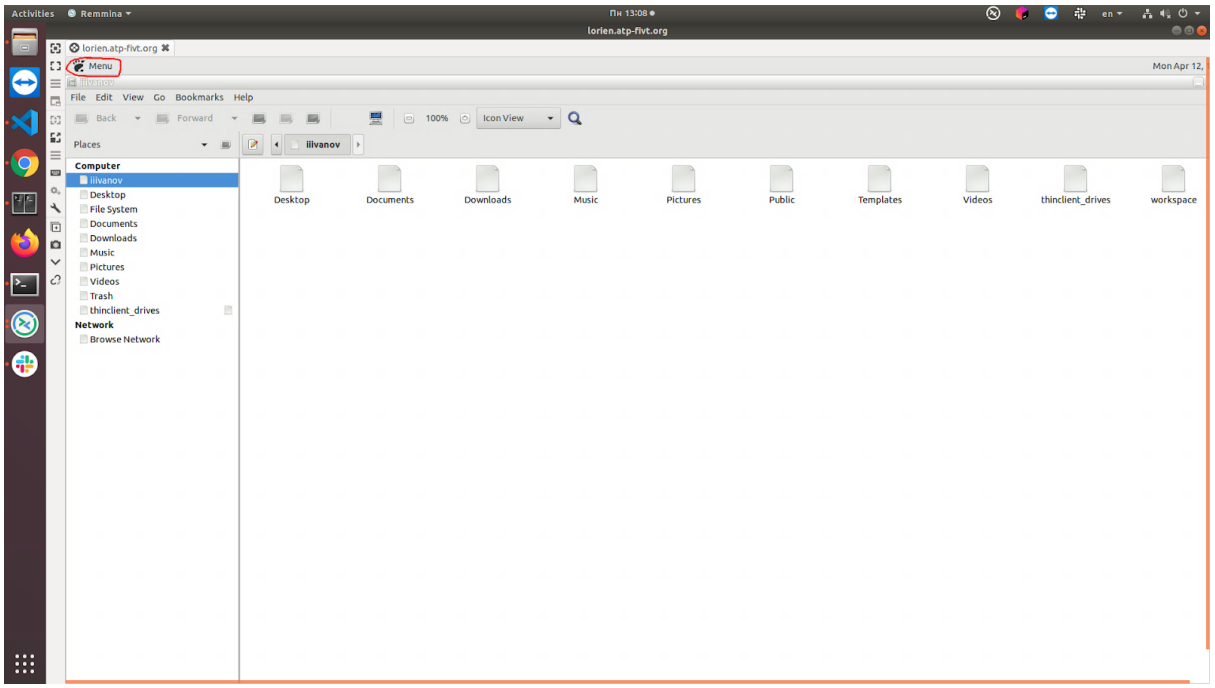
Далее сохраните настройки чтоб не вводить их при каждом подключении.
При первом подключении может вывестись ошибка:



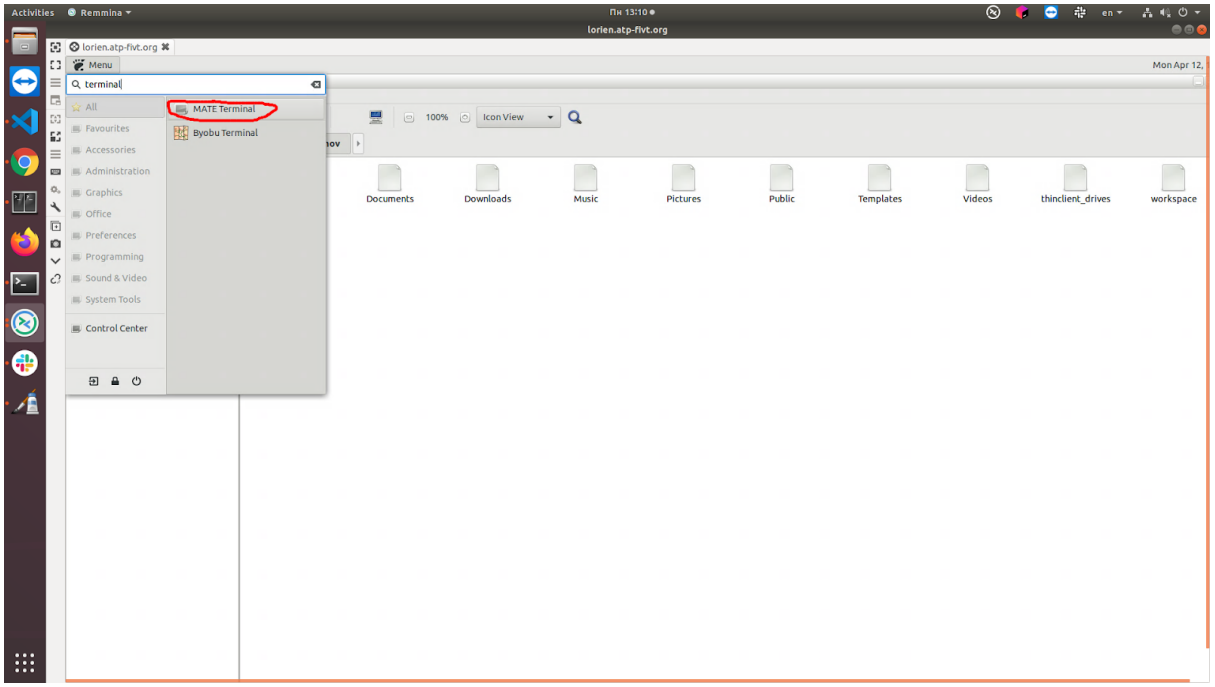
Не обращайте внимания и нажмите “Delete”.

Открытие терминала

1) Нажать на Menu, как показано обведенным красным на рисунке ниже



2) Ввести в строке поиска слово “terminal”
3) Выбрать “MATE Terminal” как показано на рисунке ниже



Дисковые квоты

На сервере есть дисковые квоты:

Диск	Квота на 1 пользователя
/	5 Gb
/home	100 Gb
/data	300 Gb
/data_fast	Квота не выдается.

Если выделенной вам квоты не хватает по уважительной причине, свяжитесь с администратором инфраструктуры через форму.

Проверить текущее состояние квоты можно с помощью команды **`quota -v`** .

Работа с Jupyter

Начало работы с Jupyter состоит из 2 этапов: запуск jupyter-сессии на сервере, подключение к серверу с локальной машины.

Важно! Не запускайте ноутбуки без пароля (токена) или со слабым паролем. Это может привести к тому, что ваш Jupyter будет взломан и злоумышленник получит доступ к ресурсам сервера через Jupyter-терминал. Про то, как устанавливать пароли для Jupyter написано здесь, а про установку токенов - см. эту команду.

При обнаружении использования серверов не по прямому назначению (не для исследований, а например для майнинга), первый раз будет выдано предупреждение, а на 2-й ваш пользователь будет отключен.

Запуск Jupyter-сессии на сервере

Чтоб Jupyter-ноутбук не падал при обрыве ssh-соединения, полезно знать об утилите tmux.

Команда	Описание
`tmux ls`	Проверка доступных tmux-сессий для вашего пользователя. Сообщение “failed to connect to server” - это не ошибка. Оно означает, что сессий, доступных вашему пользователю, нет.
`tmux new -s <name>`	Создание новой tmux-сессии.
`tmux a -t <name>`	Подключение к текущей tmux-сессии.
“Ctrl+b”, затем “d”	Выход из tmux-сессии (без её завершения).
exit	Завершение текущей tmux-сессии.

Для работы с Jupyter этих команд достаточно. Дополнительную информацию о tmux можно почитать здесь.

1. Запускаем tmux-сессию на huawei-gpu.
2. Запускаем Jupyter:

```
jupyter notebook --no-browser --port YOUR_PORT --ip 0.0.0.0
```
[1]

3. Изучаем логи JuPyter.
   a. На сервере открыты такие порты: **30000:30100**. Порт YOUR_PORT уже может быть кем-то занят и в таком случае Jupyter примется искать ближайший свободный порт. Найденный порт будет выведен в логах Jupyter после строки “*The Jupyter Notebook is running at:*”.
   b. Также стоит скопировать себе токен (ищем “*token=...*”). Этот токен потребуется ввести когда вы будете открывать Jupyter на своей машине.
   c. **Optional**. При каждом запуске Jupyter генерирует новый токен, однако этого можно избежать. Для этого создайте файл `~/.jupyter/jupyter_notebook_config.py` и запишите туда такую строку:

```
c.NotebookApp.token = u'YOUR_TOKEN'
```
[2]

   Теперь все ваши ноутбуки будут запускаться с токеном YOUR_TOKEN. Токены можно и вовсе отключить, введя вместо YOUR_TOKEN пустую строку.

Подключение к Jupyter-ноутбуку на своей машине

Подключение как для Windows, так и для Linux происходит очень просто. Достаточно набрать в браузере адрес `<HOSTNAME>:<YOUR_PORT>`. Где `<YOUR_PORT>` - это порт, на котором запущен Jupyter на сервере. Когда вы заканчиваете работу с Jupyter не забудьте корректно его завершить.

Если порт закрыт

На предыдущем шаге мы получили порт YOU_PORT, на котором работает ноутбук и токен YOUR_TOKEN, который требуется для его открытия. Если порты 30000-30100 заняты, то придётся использовать закрытый порт, пробрасывая соединение на локальную машину.

1. Вводим такую команду:

```
ssh -N -f -L YOU_PORT:localhost:YOU_PORT shad-gpu
```

Терминал больше не нужен, т.к. эта команда работает в фоновом режиме.

2. Открываем в браузере адрес: `localhost:YOU_PORT`.
3. **Optional**. После того, как вы завершили работу с ноутбуком, ssh-соединение всё равно продолжает работать. Есть 2 пути решения этой проблемы:
   a. убить соответствующий процесс, выполнив команду:

```
kill $(ps aux | grep 'localhost:YOU_PORT' | head -n -1 | cut -d ' ' -f 3)
```

   b. запускать соединение без '-f'. В таком случае терминал нужно держать открытым, но по его закрытии соединение и связанный процесс завершится.

Завершение Jupyter-сессии на сервере

1. Заходим в tmux-сессию, в которой развёрнут Jupyter.
2. Останавливаем Jupyter ("Ctrl+C", затем 'y').
3. Завершаем tmux-сессию.

Важно! Не держите tmux постоянно работающим. Как только вы закончили вычисления в Jupyter, и в ближайшее время ничего вычислять на планируйте, закрывайте tmux. В противном случае вы будете потреблять место в оперативной памяти и мешать работать другим.

Перенос данных с другого сервера

Для переброса данных из одного сервера на другой удобно использовать утилиту SCP. интаксис у SCP такой:

`scp -r <source_directory> <my_user>@<remote_ip>:<remote_directory>`

<my_user> можно не указывать если имя пользователя на исходном и удалённом серверах одинаковые. Подробнее про SCP см. здесь.

Например, вот таким образом можно перебросить данные с Server1 на Server2:

server1:~$ scp my_data server2:~/

- my_data - это папка на сервере server1,
- ~/ - папка на сервере server2 (ваш userhome),
- <my_user> - не указывается в предположении что имена пользователей у Server1 и Server2 совпадает.

При вводе SCP он будет каждый раз спрашивать пароль от удаленного сервера. Чтоб этого избежать, можно пробросить ключ с локального на удалённый сервер. Это делается с помощью команды ssh-copy-id. Используется она также, как и команда ssh. Например:

server1:~$ ssh-copy-id my_user@server2

Команда попросит пароль, но после этого вам больше не придётся вводить пароли при SSH или SCP.

Работа с библиотеками

Здесь будут описаны настройки библиотек Theano и Tensorflow, а также работающих поверх них Lasagne и Keras для работы с CPU и GPU.

Использование установленных библиотек

Conda

Внимание! В связи с некоторыми не стыковками версий драйверов, библиотеки Tensorflow, Torch и Keras были загруженны под изолированное окружение, созданное с помощью Anaconda.
Для активации работы с ними пропишите ***conda-activate***

Tensorflow

Посмотреть список доступных для tensorflow устройств (будь то CPU или GPU) можно с помощью такого кода:

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

Подключить выбранное устройство можно используя функцию `tensorflow.device('/gpu:N')`, где N - это номер GPU. Если вычисления совершаются на CPU, достаточно написать `/cpu`.

Дополнительную информацию можно прочитать здесь.

Keras

Keras может работать поверх как Tensorflow, так и Theano. По умолчанию он работает поверх Tensorflow. Если вы хотите поменять backend-библиотеку для Keras, нужно сделать следующее.

1. Сделать первый запуск, т.е. запустить любой код с Keras, чтоб библиотека сгенерировала json-конфиг.
2. Найти конфиг по пути `$HOME/.keras/keras.json` и поставить в строке `"backend"` нужный backend (возможные варианты: tensorflow, theano, cntk). Подробности здесь.

Pytorch

Pytorch обычно работает без дополнительных настроек с вашей стороны. При возникновении проблем с импортом torch, установите его себе в виртуальное окружение, выполнив:

```
pip3 install torchvision
```

Создание виртуального окружения Python

1. Для начала виртуальное окружение нужно создать, выполнив:
   ```
   python3 -m venv my_env # при этом создастся директория ~/my_env
   ```
   (вместо my_env можно поставить любое другое название)
2. Запускаем Tmux-сессию, как описано здесь. Но перед тем, как запустить собственно Jupyter, нужно войти в виртуальное окружение.
   ```
   source my_env/bin/activate
   ```
   Если всё прошло успешно, у вас перед строкой с аккаунтом появится название виртуального окружения:
   ```
   (my_env) user@shad-gpu:~$
   ```
3. Теперь можно запускать Jupyter так, как написано в разделе “Запуск Jupyter-сессии на сервере”.

Если при импорте torch возникают ошибки, убедитесь что вы работаете в ядре Python3. Если это не так, ядро можно сменить в меню ***Kernel -> Change kernel*** в Jupyter.
Для выхода из виртуального окружения достаточно просто выполнить `deactivate` находясь в нём.

Установка дополнительных библиотек на сервер

Актуальные версии основных библиотек установлены system-wide однако часто бывает так, что вам для выполнения работы нужна специфичная версия какой-либо библиотеки.

Для создания своего python-окружения можно воспользоваться `virtualenw`. Также можно использовать pip для установки библиотек в вашу пользовательскую директорию:

```
python[version] -m pip install <my_library> --user
```

Пример:

```
python3 -m pip install tensorflow==1.11 --user
```

Если что-то нужно поставить system-wide, свяжитесь с администратором инфраструктуры.

Мониторинг утилизации ресурсов GPU

На сервере развёрнута утилита NvTop. Она по аналогии с HTop позволяет мониторить нагрузку карточек. При этом чертится график утилизации GPU.

Для того, чтоб графики NvTop отрисовывался корректно, нужно:

1) Уменьшить масштаб в терминале на несколько позиций, (Ctrl + -).
2) Развернуть окно на весь экран (F11).
3) Запустить nvtop.

Автоматический киллер GPU-задач

Для выполнения "Кодекса ответственного пользователя GPU-серверов" на всех серверах кроме shad-gpu, запущена система автоматической остановки задач, которая работает по следующему принципу.

Каждые 2 часа запускается анализатор выдачи nvidia-smi. Он формирует чёрный список PID таким образом:

- если обнаружено, что какая-то карточка простаивает (<5 % util и > 90% загрузка памяти), все процессы на ней добавляются в список.
- если обнаружено, что у какого-то пользователя превышено кол-во используемой видеопамяти (больше 12Gb суммарно на всех карточках), его PID'ы тоже добавляются в чёрный список (обычно у 1 пользователя запущена 1 программа и, следовательно, только 1 PID).

Через 5 мин. проверка повторяется т.к. утилизация у программы может поменяться.

По результатам этих 2 проверок:

- чёрный список сохраняется в файл,
- владельцам PID'ов из списка высылаются письма с предупреждениями.

При следующем прогоне анализатора процессы из списка будут убиты, а их PID'ы и владельцы зафиксированы в лог-файле.

Кодекс ответственного пользователя инфраструктуры

Как вы могли убедиться, на серверах ШАД одновременно работает много пользователей поэтому важно следить, сколько ресурсов потребляет ваша задача. Для упрощения жизни ваших коллег просьба соблюдать несколько простых шагов.

1. Не оставляйте висеть задачи, которые уже ничего не вычисляют поскольку они лочат память видеокарты. Проверить активность вашей задачи можно с помощью столбца ***"Volatile uncorr. ecc gpu-util compute"*** в выводе команды ***nvidia-smi***. Если вы видите, что память на карточке занята, а Volatile ecc. долгое время 0%, проверьте, не посчиталась ли ваша задача.

2. Используйте переменную окружения **CUDA_VISIBLE_DEVICES**. С помощью неё можно задать, на какой (-их) карточке будет работать ваша задача. Пример:

   ```
   CUDA_VISIBLE_DEVICES=0,1 jupyter notebook…
   ```

   Обычные задачи ШАД чаще всего не требуют больше одной видеокарты.

   В коде эту переменную можно задать с помощью `os.environ`. Например:

   ```
   os.environ['CUDA_VISIBLE_DEVICES'] = '0,1'
   ```

3. Чтобы ограничить распараллеливание numpy на надо выставить переменную окружения **OPENBLAS_NUM_THREADS**.

```
import os
# Переменную окружения выставить ДО импорта numpy и sklearn
# Количество использованных ядер = n_jobs * OPENBLAS_NUM_THREADS
# Поэтому, если используете n_jobs > 1, то, пожалуйста, ставьте OPENBLAS_NUM_THREADS
= 1
os.environ["OPENBLAS_NUM_THREADS"] = "1"
```

4. В tensorflow есть возможность задать, какую долю видеопамяти на каждой из выделенных карточек будет использовать ваша задача.

   ```
   conf = tf.ConfigProto() # default config
   conf.gpu_options.per_process_gpu_memory_fraction = 0.3
   ```

   Настроенную конфигурацию Tensorflow можно также подключить к Keras:

   ```
   K.set_session(tf.Session(graph=tf.get_default_graph(), config=conf))
   ```

   Для того, чтоб получить актуальные данные о загруженности кластера, выполните команду **`nvidia-smi`**.

5. Также есть возможность ещё более гибко расходовать память:

   ```
   config.gpu_options.allow_growth = True
   ```

   С помощью этой настройки TF будет расходовать столько GPU-памяти, сколько ей нужно в данный момент.

6. В Tensorflow 2.x API для управления долями GPU перенесли в experimental, но им можно воспользовтся 2м способами:

```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.2
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

```
import tensorflow as tf
gpus = tf.config.list_physical_devices('GPU')

tf.config.experimental.set_memory_growth(gpus[0], True)
tf.config.experimental.set_virtual_device_configuration(
            gpus[0],
            [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=1024)]
)
```

Подробнее см. здесь.

7. Не ставьте слабые пароли на свои Jupyter-ноутбуки (и тем более не отключайте пароли) чтоб к вашему Jupyter не могли подключиться злоумышленники. Подробнее см. здесь.
8. Также не забывайте иногда смотреть на мониторинг утилизации GPU-ресурсов.

Просьба выполнять эти несложные предписания. В противном случае сработает программа, которая будет **автоматически убивать процессы на GPU** после определённого времени работы.

Контакты

В случае проблем с инфраструктурой, обратитесь к администратору через эту форму. Если у вас нетипичная проблема, для которой не предусмотрены поля в форме - обратитесь на **servers@atp-fivt.org**.

Q & A

1. **Вопрос.** Пытаюсь установить библиотеку с помощью pip и вижу:
   ```
   Could not install packages due to an EnvironmentError: [Errno 122] Disk quota exceeded
   ```
   При этом `quota -v` показывает, что место есть.
   Ответ. При установке пакетов pip кеширует данные в /tmp. Это расходует квоту в ‘/’ и хотя файлы и удалятся после установки, в момент установки квоты может не хватить. Для этого нужно явно указать папку для кеша на другом диске. Подробнее здесь.
2.