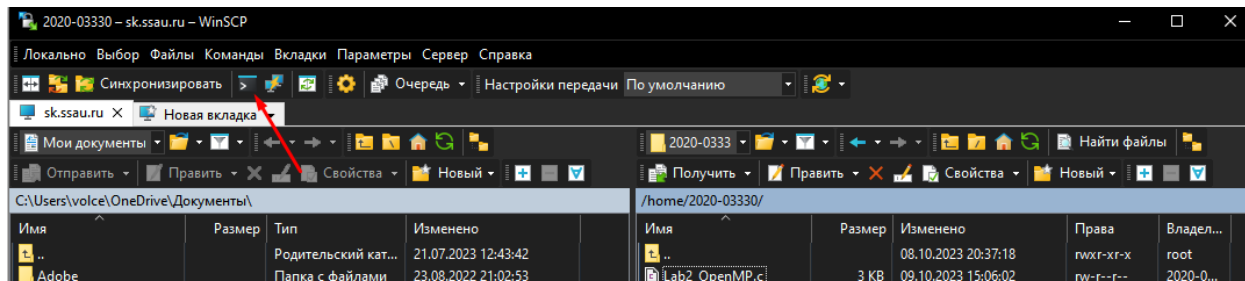


Здесь будет рассмотрен запуск на примере [WinSCP](#)

1. Подключаемся к кластеру, используя выданные логин и пароль.
2. Создаём файл название-с.с (файл C)/название-cu.cu (файл CUDA).
3. Открываем консоль



4. Компилируем файлы

4.1. Для OpenMP:

1. Прописываем `module load intel/icc18` – таким образом мы подгрузили нужные модули.
2. Для компиляции программы прописываем `gcc -fopenmp -o test название-с.с`. Здесь название-с.с – созданный нами файл, а test – название выходного. Обратите внимание, что у выходного файла не указывается расширение.

4.2 Для MPI:

1. Прописываем `module load intel/mpi4` – таким образом мы подгрузили нужные модули.
2. Для компиляции программы прописываем `mpicc -o test название-с.с`. Здесь название-с.с – созданный нами файл, а test – название выходного. Обратите внимание, что у выходного файла не указывается расширение.

4.3 Для последовательной:

1. Прописываем `module load intel/icc18` – таким образом мы подгрузили нужные модули.
2. Для компиляции программы прописываем `gcc -o test название-с.с`. Здесь название-с.с – созданный нами файл, а test – название выходного. Обратите внимание, что у выходного файла не указывается расширение.

5. После того, как прописали команду, закрываем консоль.
6. Создаём скрипт: ПКМ → Новый → Файл. Создаём файл название.sh.
7. Прописываем в открывшемся файле скрипт.

7.1. Для OpenMP:

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --time=00:03:00
#SBATCH --nodes=1 --ntasks-per-node=N
#SBATCH --mem=1gb

./test
```

7.2. Для MPI:

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --time=00:03:00
#SBATCH --nodes=1 --ntasks-per-node=N
#SBATCH --mem=1gb

export I_MPI_LIBRARY=/usr/lib64/slurm/mpi_pmi2.so
srun --mpi=pmi2 ./test
```

7.3 Для последовательной:

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --time=00:03:00
#SBATCH --nodes=1
#SBATCH --mem=1gb

./test
```

В строчке `--ntasks-per-node=N` `N` – желаемое количество процессов для одной ноды, `./test` – путь до файла, созданного нами после компиляции. В строке `--mem=1gb` указывается количество памяти, которое будет выделено под вашу программу.

7.4 Для CUDA (компиляция и запуск в одном):

```
#!/bin/bash

#SBATCH --job-name=myjob
#SBATCH --nodes=1
#SBATCH --gres=gpu
#SBATCH --time=00:03:00

module load cuda/8.0
nvcc -g -G -O0 -DGRID_SIZE=<GRID_SIZE> -DBLOCK_SIZE=<BLOCK_SIZE>
-DNMAX=<NMAX> -lcublas -o название-cu.bin название-cu.cu
./название-cu.bin
```

Параметры `-DGRID_SIZE`, `-DBLOCK_SIZE` и `-DNMAX` – необязательные. Вы можете их указать (после равно нужно указать свои значения), можете нет. `-D` заменяет, по сути, `#define`. Доступ к переменным в коде будет как обычно – будто они объявлены. Если вы решите не прописывать эти параметры, тогда переменные `GRID_SIZE`, `BLOCK_SIZE` и `NMAX` должны быть объявлены явно.

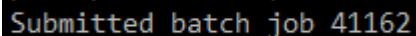
Также название `-cu.bin` – название выходного файла. Назвать его можно как угодно, только не забудьте поменять его и в строке `./название-cu.bin`.

В строчке `--job-name=myjob myjob` `myjob` – имя вашей операции, которое будет отображаться в очереди – может быть любым. Строчка `--time=00:03:00` отвечает за то, сколько времени отведётся для выполнения вашей программы на кластере. А в строчке с `--nodes=M` указывается число нод, на которых будет выполняться ваша программа (учтите, что когда запускаете программу на нескольких нодах и указываете в скрипте через `--ntasks-per-node=N` количество процессов для каждой ноды, в сумме у вас программа запустится на $M \cdot N$ процессах)

8. Сохраняем файл и закрываем его.

9. Открываем консоль и прописываем в ней `sbatch название.sh`, где `название.sh` – созданный скрипт для запуска программы.

10. После выполнения скрипта в консоли должно появиться сообщение следующего типа:



```
Submitted batch job 41162
```

Это номер нашего скрипта, запущенного в работу. Просмотреть очередь можно с помощью команды `squeue` или `squeue -l`, которая покажет более подробную информацию.

11. Закрываем консоль и видим, что в папке после запуска скрипта появился файл `slurm-номер.out`, где хранится весь консольный вывод программы.

12. Ждём окончания выполнения программы.