Подготовка к работе со стандартом OpenACC

САМОСТОЯТЕЛЬНАЯ РАБОТА

http://www.kimrt.ru
Онлайн-курс
Суперкомпьютерные технологии в задачах моделирования

OpenACC (от англ. Open Accelerators) — программный стандарт для параллельного программирования, разрабатываемый совместно компаниями Cray, CAPS, Nvidia и PGI. Стандарт описывает набор директив компилятора, предназначенных для упрощения создания гетерогенных параллельных программ, использующих как центральный, так и графический процессор [1].

Конфигурация используемого в текущей работе ПК:

- процессор Intel Core i7- 8700;
- материнская плата MSI B365M PRO-VDH;
- ОЗУ DDR4 32 GB dual channel;
- видеокарта NVIDIA RTX 2060 super.

1. Установка ОС

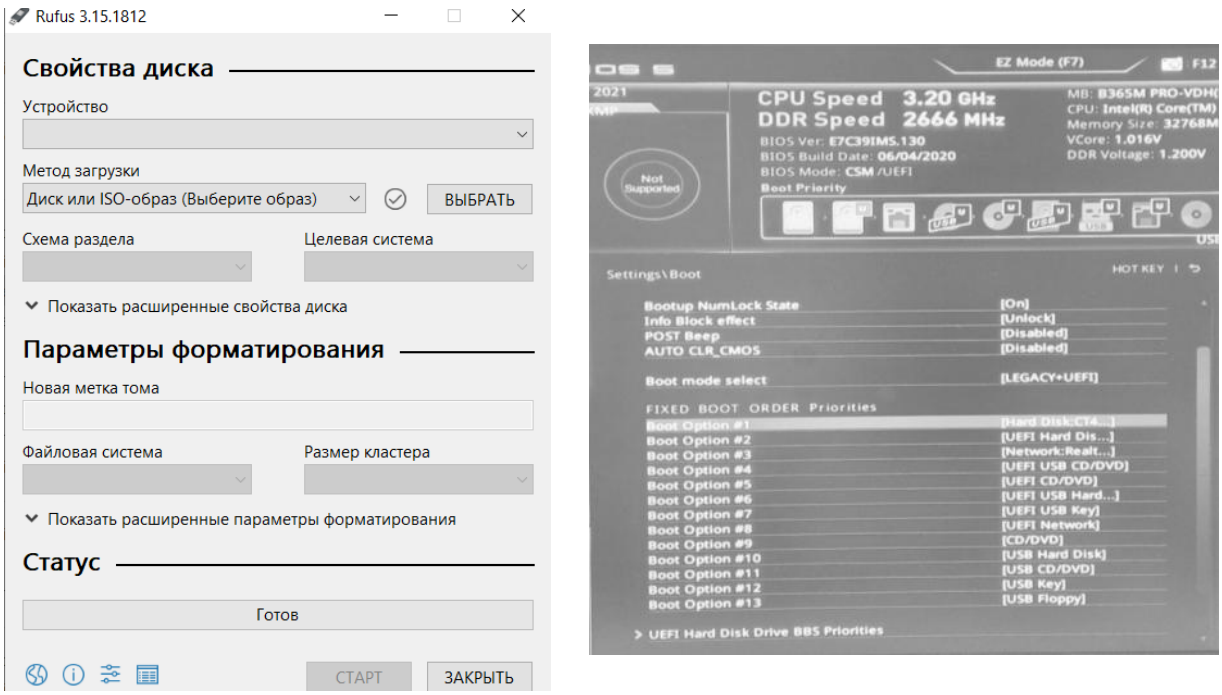

Рис. 1. Приложение для создания загрузочных USB-накопителей (слева) и установка приоритета загрузочных дисков (справа)

Для начала необходимо установить операционную систему из семейства Linux, например, CentOS 8. В текущем примере CentOS ставилась с

графическим интерфейсом второй системой, после Windows 10, на отдельный жесткий диск. Для установки ОС следует выполнить следующие действия. Загрузите ISO образ с официального сайта (https://www.centos.org/). Затем создайте загрузочный USB-флеш-накопитель или диск (рис. 1, слева).

Выставляем приоритет загрузки в BIOS или UEFI с накопителя или диска (рис. 1, справа). Устанавливаем операционную систему (рис. 2). В нашем случае CentOS 8, ядро: 4.18.0-305.3.1.e18.x86_64.

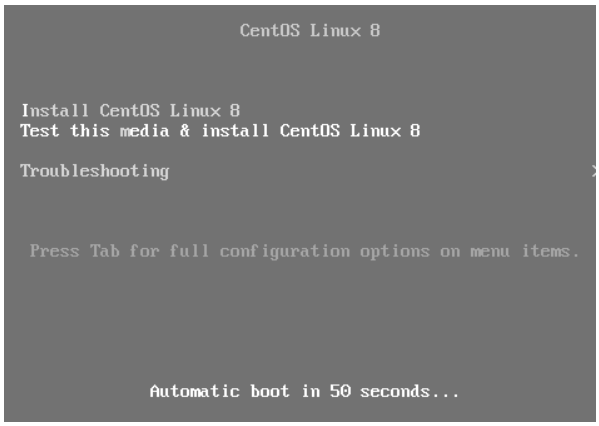

Рис. 2. Установка CentOS

2. Установка драйвера Nvidia

Чтобы использовать видеокарту для вычислений, необходимо установить драйвер от Nvidia вместо имеющегося по умолчанию драйвера Nouveau [2]. Выполняем следующую команду для отключения Nouveau [3].

```
# sudo vi /etc/modprobe.d/blacklist-nouveau.conf
```

Нажимаем i для перехода в режим редактирования и записываем в конец файла следующие строки (рис. 3, слева).

```
blacklist nouveau
options nouveau modeset=0
```

Для выхода из режима редактирования файла нажимаем Esc, для сохранения и выхода выполняем следующую команду.

```
:wq
```

Когда вводится двоеточие, курсор переходит на последнюю строку экрана, и таким образом редактор оказывается в режиме последней строки. Команда w сохраняет файл, а команда q закрывает его.

Далее выполняем следующие команды, чтобы перезаписать существующий файл initramfs.

```
# sudo dracut --force
# reboot
```

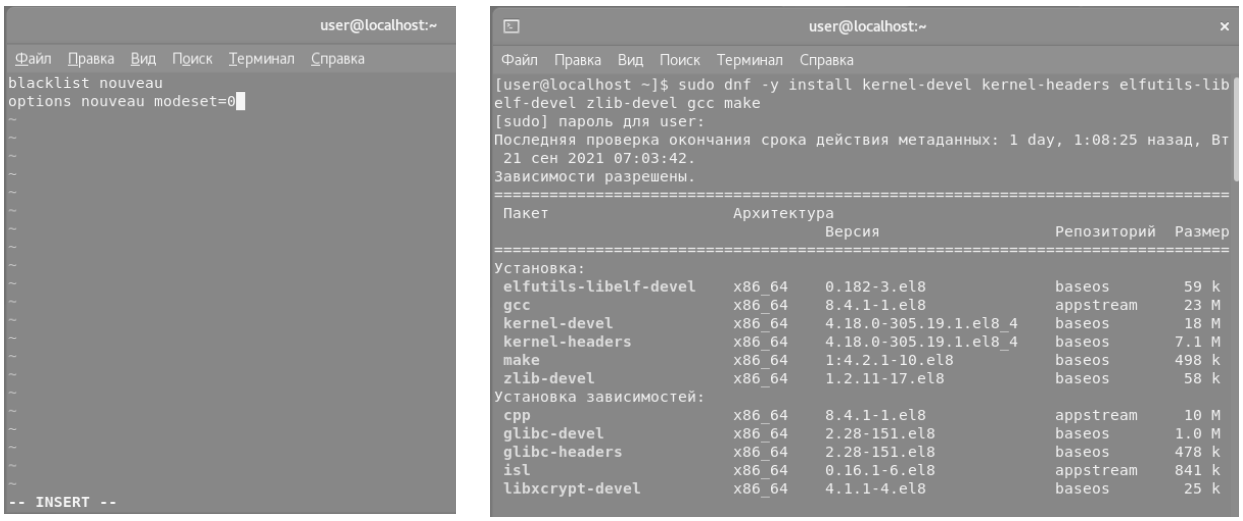

Рис. 3. Редактор vi (слева) и установка пакетов (справа)

После чего произойдет перезагрузка системы. Необходимо установить некоторые пакеты.

```
# sudo dnf -y install kernel-devel kernel-headers elfutils-libelf-devel zlib-devel gcc make
```

Скачиваем драйвер с официального сайта Nvidia для вашей видеокарты (рис. 4, слева).

Рис. 4. Загрузка драйвера с сайта NVIDIA (слева) и режим CUI (справа)

Если вы используете GUI, то необходимо перейти в режим CUI (рис. 4, справа). Существует несколько способов изменения режима. Воспользуемся одним из них и выполним в терминале следующую команду [4].

```
# sudo systemctl isolate multi-user.target
```

Далее необходимо войти в качестве пользователя, введя логин и пароль. Если строка ввода логина не появилась, попробуйте нажать alt+F3. Если у вас вместо кириллицы отображаются квадраты, необходимо изменить шрифт [5]. В файле /etc/vconsole.conf изменяем переменную FONT таким образом: FONT="UniCyr_8x16". Затем переходим в папку с драйвером.

```
# cd Загрузки
```

Чтобы посмотреть название файла драйвера, выводим список файлов.

```
# ls
```

Запускаем установку драйвера.

```
# bash NVIDIA-Linux-x86_64-430.50.run
```

При установке возможно возникновение ошибки “Unable to find the kernel source tree for currently running kernel” [6]. Для исправления ошибки и установки правильных заголовочных файлов ядра выполните команду ниже.

```
# sudo yum install "kernel-devel-uname-r == $(uname -r)"
```

На этапе установки драйвера отвечаем *Yes* на вопрос 1) “*Install NVIDIA's 32-bit compatibility libraries?*” и *No* на вопрос 2) “*Would you like to run the nvidia-xconfig utility to automatically update your X configuration file so that the NVIDIA X driver will be used when you restart X? Any pre-existing X configuration file will be backed up*”.

Для проверки корректности установки драйвера выполним следующую команду.

```
# nvidia-smi
```

После этого должно появиться следующее окно (рис. 5).

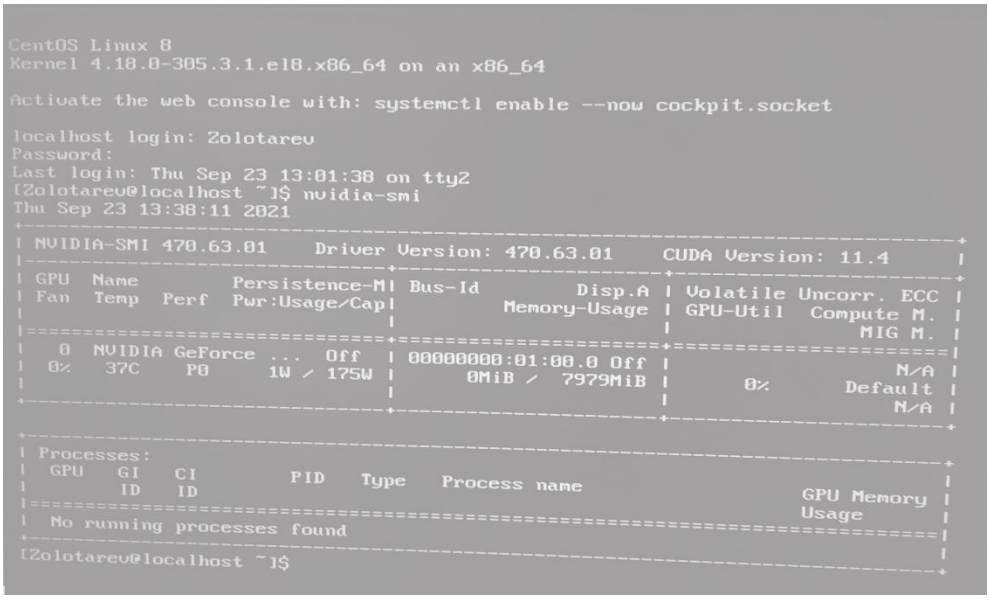

Рис. 5. Параметры видеокарты и драйвера

Перезагружаем систему.

```
# reboot
```

3. Установка hpc sdk

Загружаем необходимые программные наборы [7–9].

```
# wget https://developer.download.nvidia.com/hpc-sdk/21.7/nvhpc-21-7-21.7-1.x86_64.rpm
# wget https://developer.download.nvidia.com/hpc-sdk/21.7/nvhpc-2021-21.7-1.x86_64.rpm
```

Затем устанавливаем их.

```
# sudo yum install ./nvhpc-21-7-21.7-1.x86_64.rpm
./nvhpc-2021-21.7-1.x86_64.rpm
```

Теперь необходимо прописать компиляторы в *PATH* [10]. Создадим в *etc/profile.d* файл, например, *nvidiahpcsdk.sh*. Это можно сделать следующим образом. Перейдем в каталог *etc/profile.d* с помощью команды *cd*. Создадим необходимый файл.

```
# sudo vi nvidiahpcsdk.sh
```

Запишем в него команду экспорта. Команда export предназначена для экспорта переменных и функций текущего процесса в дочерний процесс.

```
# export PATH=”$PATH:/opt/nvidia/hpc_sdk/
Linux_x86_64/21.7/compilers/bin”
```

После этого сохраним файл.

Чтобы *$PATH* обновился, выполним следующую команду.

```
# . /etc/profile
```

Для проверки работоспособности компилятора выполним команду (рис. 6).

```
# nvaccelinfo
```

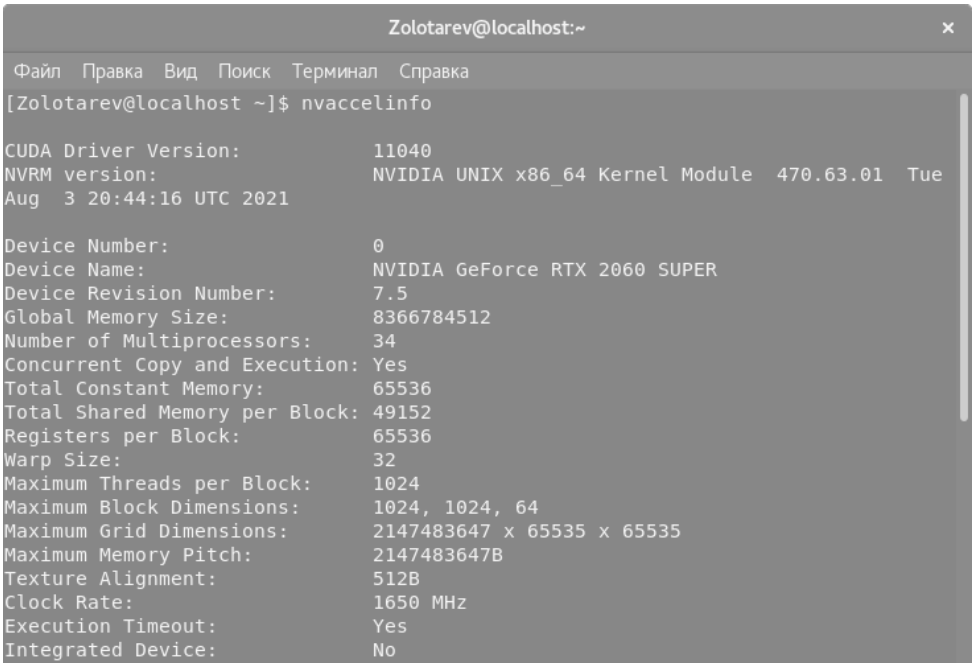

Рис. 6. Конфигурация SDK

4. Тестирование OpenACC

Для тестирования работоспособности OpenACC необходимо загрузить программу с GitHub [11]. В этой программе решается двумерное уравнение Лапласа

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

с граничным условием $u|_\Gamma = \varphi$ итерационным методом Якоби [12]. Здесь $u(x,y)$ – искомая функция и φ – какая-то константа (значение u на границе области). Запишем разностную схему «крест» для уравнения Лапласа

$$\frac{u_{i-1,j}^k - 2u_{i,j}^{k+1} + u_{i+1,j}^k}{h^2} + \frac{u_{i,j-1}^k - 2u_{i,j}^{k+1} + u_{i,j+1}^k}{h^2} = 0,$$

где h – пространственный шаг между узлами сетки, k – номер итерации, i, j – позиции узла в столбце и строке, соответственно. Далее получаем выражение

$$u_{i,j}^{k+1} = \frac{1}{4}\left(u_{i-1,j}^k + u_{i+1,j}^k + u_{i,j-1}^k + u_{i,j+1}^k\right)$$

для итерационного расчета. В листинге 1 представлен код программы на языке C++, реализующий данный метод.

Листинг 1. Решение уравнения Лапласа методом Якоби

```
#include <math.h>
#include <string.h>
#include "timer.h"

#define NN 4096
#define NM 4096

double A[NN][NM];
double Anew[NN][NM];

int main(int argc, char** argv)
{
    const int n = NN;
    const int m = NM;
    const int iter_max = 1000;

    const double tol = 1.0e-6; // точность расчета
    double error     = 1.0;

// заполняем массивы нулями
    memset(A, 0, n * m * sizeof(double));
    memset(Anew, 0, n * m * sizeof(double));
// задаем граничное условие
    for (int j = 0; j < n; j++)
    {
        A[j][0]    = 1.0;
        Anew[j][0] = 1.0;
    }

    printf("Jacobi relaxation Calculation: %d x %d mesh\n", n, m);
```

```
    StartTimer();
    int iter = 0;

#pragma acc data copy(A), create(Anew) //выполняем итерации
    while ( error > tol && iter < iter_max )
    {
        error = 0.0;

#pragma omp parallel for shared(m, n, Anew, A)
#pragma acc kernels
        for( int j = 1; j < n-1; j++)
        {
            for( int i = 1; i < m-1; i++ )
            {
                Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1]
                                    + A[j-1][i] + A[j+1][i]);
                error = fmax( error, fabs(Anew[j][i] - A[j][i]));
            }
        }

#pragma omp parallel for shared(m, n, Anew, A)
#pragma acc kernels
        for( int j = 1; j < n-1; j++)
        {
            for( int i = 1; i < m-1; i++ )
            {
                A[j][i] = Anew[j][i];
            }
        }

        if(iter % 100 == 0) printf("%5d, %0.6f\n", iter, error);

        iter++;
    }

    double runtime = GetTimer();

    printf(" total: %f s\n", runtime / 1000);
}
```

Зайдем в директорию, где расположена программа с помощью команды cd, и выполним компиляцию [13].

```
# pgcc –acc -o laplace2dacc laplace2d.c
```

Теперь запускаем ее.

```
# ./laplace2dacc
```

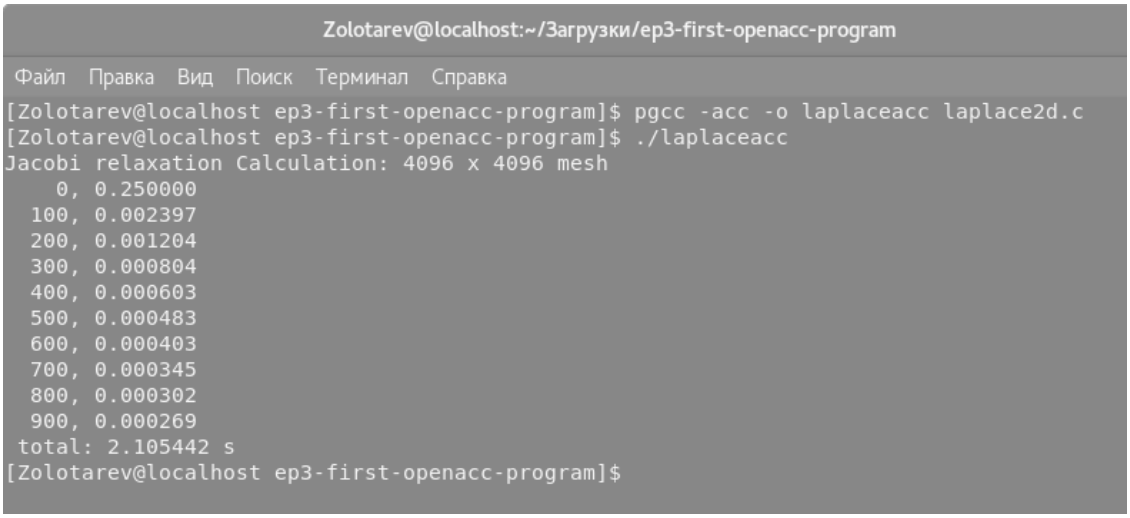

Рис. 7. Тестирование OpenACC

В окно консоли выводится номер итерации и максимальная погрешность (рис. 7). Массивы *A* и *Anew* хранят значения *u* на предыдущей и текущей итерации. Программа выполнилась за 2.1 с. Для сравнения, эта же программа на одном потоке CPU выполнялась за 174.35 с. С принципом распараллеливания алгоритма в этой задаче предлагается разобраться самостоятельно [14].

http://www.kimrt.ru/index/course_stm/0-24

Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина.

Источники

1. Стандарт OpenACC. URL: https://ru.wikipedia.org/wiki/OpenACC
2. Установка драйвера NVIDIA. URL: https://www.server-world.info/en/note?os=CentOS_8&p=nvidia
3. Работа с редактором vi. URL: https://docs.altlinux.org/ru-RU/archive/2.3/html-single/junior/alt-docs-extras-linuxnovice/ch02s10.html
4. Смена runlevel. URL: https://www.tecmint.com/change-runlevels-targets-in-systemd/
5. Квадраты вместо кириллицы. URL: https://sysadminfaq.ru/linux/centos-8.-kvadratyi-vmesto-russkix-simvolov-v-konsoli
6. Неправильный заголовочный файл ядра. URL: https://qastack.ru/unix/110682/yum-installs-kernel-devel-different-from-my-ernel-version
7. NVIDIA HPC-SDK. URL: https://developer.nvidia.com/hpc-sdk

8. Загрузка NVIDIA HPC-SDK. URL: https://developer.nvidia.com/nvidia-hpc-sdk-downloads
9. Установка HPC-SDK. URL: https://docs.nvidia.com/hpc-sdk/compilers/openacc-gs/index.html
10. Добавление директории в PATH. URL: https://qastack.ru/server/102932/adding-a-directory-to-path-in-centos
11. OpenACC программа. URL https://github.com/NVIDIA-developer-blog/cudacasts/tree/master/ep3-first-openacc-program
12. Численные методы решения уравнений в частных производных. URL: https://intuit.ru/studies/courses/1170/213/lecture/5499?page=7
13. CUDACast #3 - Your First OpenACC Program (video). URL: https://www.youtube.com/watch?v=_do2Dwa29EM
14. OpenACC Fundamentals. URL: http://icl.cs.utk.edu/classes/cosc462/2017/pdf/OpenACC_2.pdf