Вычисления на CPU средствами Matlab Parallel Computing Toolbox: параллельный цикл parfor

САМОСТОЯТЕЛЬНАЯ РАБОТА

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

Подключение библиотеки Parallel Computing Toolbox к Matlab. Чтобы начать параллельные вычисления в Matlab, необходимо установить специальное расширение (библиотеку) Parallel Computing Toolbox [1]. Для установки находим его в поисковике (рис. 1).

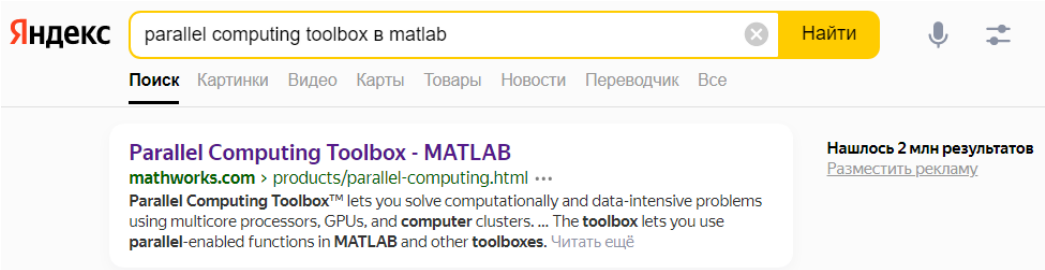

Рис. 1. Находим Parallel Computing Toolbox в браузере

Проходим по первой ссылке (сейчас для этого может быть необходимо включить VPN, например Browsec) и оказываемся на главной странице этой библиотеки (рис. 2).

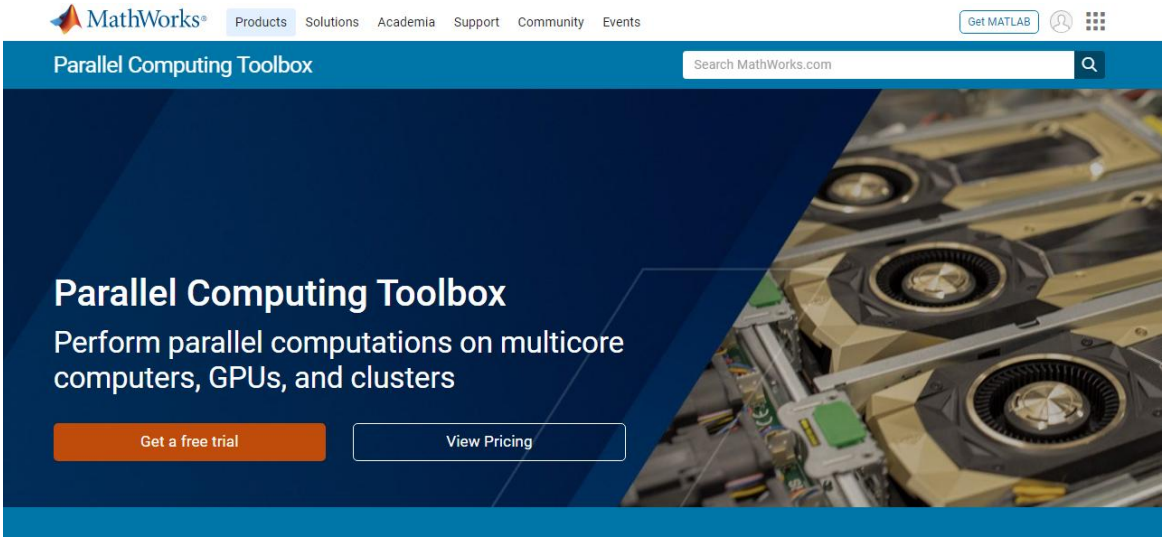

Рис. 2. Сайт расширения Parallel Computing Toolbox

Необходимо зайти в свой Mathworks аккаунт (иконка в правом верхнем углу) и войти. Появится окно входа (рис. 3).

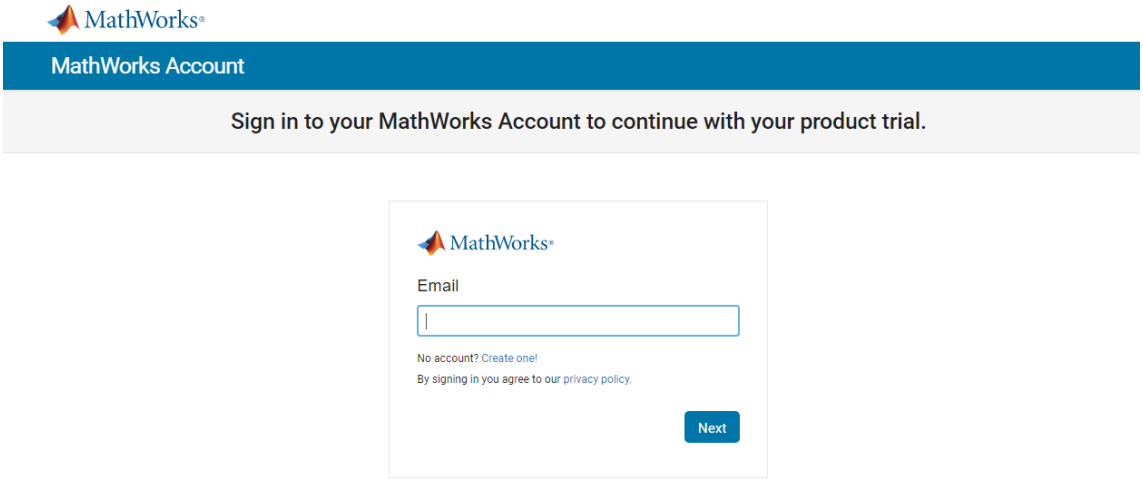

Рис. 3. Окно входа в Mathworks аккаунт

Входим в Mathworks аккаунт и нажимаем кнопку «Get a free trial». Далее устанавливается расширение Parallel Computing Toolbox и в нижнем левом углу Matlab появляется специальный блок параллельных вычислений (рис.4). При нажатии на кнопку в виде progress bar мы можем увидеть меню.

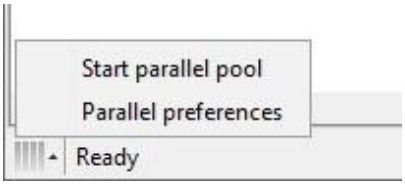

Рис. 4. Меню блока параллельных вычислений

При нажатии на вариант «Parallel preferences» в выплывающем меню (рис.4) открывается окно «Parallel Computing Tollbox Preferences» (рис.5), где можно настроить работу Parallel Computing Toolbox. Можно выбрать кластер для вычислений (на ноутбуке или стационарном личном компьютере доступен только локальный кластер) и предпочтительное количество ядер (workers) для распараллеливания. Также можно включить автоматический запуск параллельного пула (это и есть фактически запуск режима параллельных вычислений) по ключевым словам (например, *parfor*) с помощью кнопки в виде check box (обычно этот выбор стоит по умолчанию). Также можно выбрать отключение (через определённое

время) режима параллельных вычислений (по умолчанию этот выбор тоже стоит и время, после которого режим отключается стоит 30 минут).

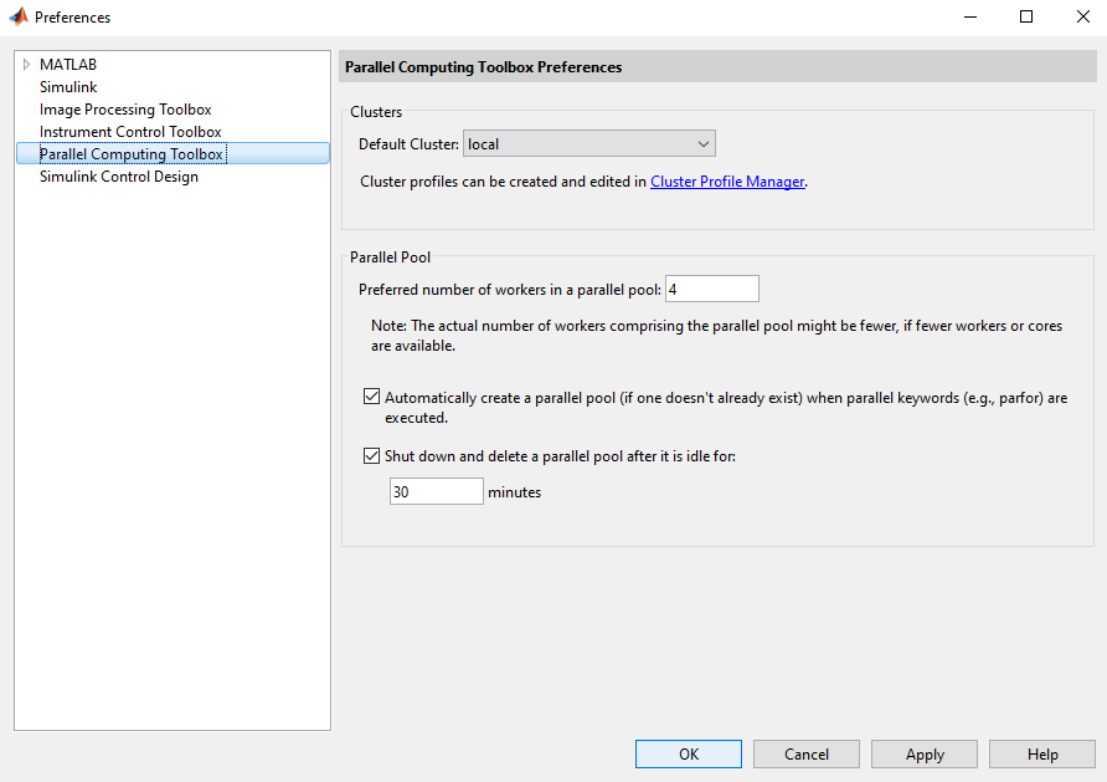

Рис. 5. Панель элемента Parallel Preferences

Чтобы включить параллельные вычисления необходимо выбрать элемент «Start Parallel pool» и дождаться, пока progress bar загрузится (по умолчанию расчёт идёт на 4 ядрах или же на всех доступных).

Примеры параллельных вычислений. Теперь рассмотрим примеры ускорения работы программ с помощью распараллеливания [2]. Создадим цикл *parfor* для задачи, требующей больших вычислительных ресурсов, и измерим полученное ускорение. В редакторе Matlab введём следующий цикл *for*. Чтобы измерить прошедшее время, добавим *tic* и *toc*.

```
tic
n = 200;
A = 500;
a = zeros(1,n);
```

```
for i = 1:n
    a(i) = max(abs(eig(rand(A))));
end
toc
```

Теперь запустим этот код и посмотрим, сколько он выполнялся (процессор AMD A6- 6310 APU - 4 ядра; 2,4 ГГц)

```
Elapsed time is 103.838126 seconds.
```

В данном примере используется функция `rand(A)`, которая создаёт матрицу размерностью A x A, заполненную случайными числами. Затем для полученной матрицы находится вектор собственных значений с помощью функции `eig()`. После этого находится модуль каждого элемента вектора с помощью функции `abs()` и из полученных модулей выбирается наибольшее значение с помощью функции `max()`. Также используется функция `zeros()`, которая позволяет инициализировать матрицу или вектор (в данном случае вектор) нулями.

Теперь заменим цикл *for* на *parfor*

```
tic
n = 200;
A = 500;
a = zeros(1,n);
parfor i = 1:n
    a(i) = max(abs(eig(rand(A))));
end
toc
```

Запустим новый скрипт и затем запустим его повторно. Первый запуск медленнее второго, потому что должен быть запущен параллельный пул или режим параллельных вычислений. Запуск этого режима переводит Matlab в состояние готовности делать параллельные вычисления и код становится доступным для вычислений на нескольких ядрах. После этого тут же начинаются вычисления. Именно из-за того, что при первом запуске сначала происходит подготовка, он и занимает значительно больше времени, чем последующие. Посмотрим, какое время занял второй запуск.

```
Elapsed time is 73.239400 seconds.
```

Вычисления проходили на двух ядрах (workers). По умолчанию Matlab запускает вычисления на всех доступных на данном компьютере ядрах.

Обратите внимание, мы ускорили вычисления, преобразовав цикл *for* в цикл *parfor* для двух ядер. Можно еще сильнее сократить затраченное время, увеличив количество ядер в parallel pool. Для изменения количества ядер нужно будет зайти в Parallel Preferences и изменить поле “Preferred number of workers in a parallel pool”. Например, если увеличить количество ядер до четырёх мы получим ещё большее ускорение.

```
Elapsed time is 49.284198 seconds.
```

Ограничения при распараллеливании. В некоторых случаях [3] невозможно распараллелить код или необходимо сильно изменить код для преобразования циклов *for* в циклы *parfor*. В этом примере показано, какие проблемы возникают с циклом *parfor,* который содержит вложенный цикл *parfor*. Попробуем распараллелить этот

код (в данном случае переменная x не влияет на массив A(y), вложенные циклы нужны для примера).

```
for x = 0:0.1:1
    for y = 2:10
        A(y) = A(y-1) + y;
    end
end
```

Для этого попробуем преобразовать циклы *for* в циклы *parfor*. Обратите внимание, что этот код выдает ошибки (рис. 6).

```
parfor x = 0:0.1:1
    parfor y = 2:10
        A(y) = A(y-1) + y;
    end
end
```

В этом случае мы не можем просто преобразовать циклы *for* в циклы *parfor* без каких-либо других изменений. Это связано с детерминированным порядком заполнения массива , иными словами – с зависимостью элементов массива по данным.

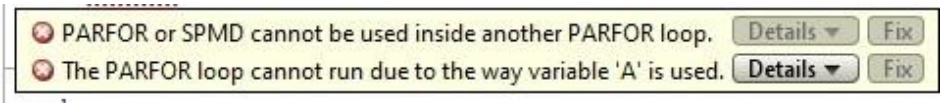

Рис. 6. Примеры ошибок при неправильном использовании *parfor*

Чтобы решить эти проблемы, необходимо изменить код. Тело цикла *parfor* выполняется в parallel pool с использованием нескольких ядер и в недетерминированном порядке. Следовательно, необходимо следующие требования к телу цикла *parfor*:

1) Тело цикла *parfor* должно быть независимым. Одна итерация цикла не может зависеть от предыдущей итерации, потому что итерации выполняются параллельно в недетерминированном порядке. В примере

```
A(y) = A(y-1) + y;
```

не является независимым, и поэтому мы не можем использовать *parfor*.

2) Мы не можем вложить цикл *parfor* в другой цикл *parfor*. В примере есть два вложенных цикла *for*, поэтому мы не можем заменить только один цикл *for* циклом *parfor*. Вместо этого вы можете вызвать функцию, которая использует цикл *parfor* внутри тела другого цикла *parfor*. Однако такие вложенные циклы *parfor* не дают вычислительных преимуществ, поскольку все ядра используются для распараллеливания самого внешнего цикла.

3) Переменные цикла *parfor* (их ещё называют счётчиками) должны быть последовательными возрастающими целыми числами. В примере

```
parfor x = 0:0.1:1
```

имеет нецелочисленные переменные цикла, поэтому здесь нельзя использовать *parfor*. Вы можете решить эту проблему, перестроив алгоритм так, чтобы переменные цикла были целочисленными.

4) Вы не можете выйти из цикла *parfor* раньше, как это можно было в цикле *for*. Не включайте оператор *return* или *break* в тело вашего цикла *parfor*.

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

Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры 2020/2021 благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина.

Источники

1. Parallel Computing Toolbox - MATLAB [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/products/parallel-computing.html, – (дата обращения: 10.04.2022).
2. Execute for-loop iterations in parallel on workers - MATLAB parfor [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/help/parallel-computing/parfor.html, – (дата обращения: 15.04.2022).
3. Convert for-loops into parfor-loops - MATLAB [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/help/parallel-computing/convert-for-loops-into-parfor-loops.html, – (дата обращения: 15.04.2022).