Модель параллельного программирования Task в Maple

САМОСТОЯТЕЛЬНАЯ РАБОТА

http://www.kimrt.ru

Онлайн-курс

Суперкомпьютерные технологии в задачах моделирования

Модель программирования Task – это модель программирования высокого уровня, предназначенная для упрощения многопоточного программирования. В этой модели пользователи создают задачи вместо потоков. Task – это вызов функции (процедура с аргументами), которая может выполняться в потоке. Maple автоматически распределяет задачи по потокам. Это позволяет масштабировать код, написанный с использованием модели программирования задач, для компьютеров с большим количеством процессоров.

Maple включает модель программирования Task, которая реализуется библиотекой параллельного программирования высокого уровня. Модель программирования Task позволяет пользователям реализовывать параллельные алгоритмы, способные использовать преимущества многоядерных компьютеров, которые сейчас широко распространены. Написание параллельных алгоритмов с использованием модели программирования Task уменьшает и устраняет многие трудности, связанные со стандартным многопоточным программированием [2].

Вот несколько преимуществ использования модели программирования Task.

•Нет явных потоков, так как пользователи создают задачи, а не потоки.

•Maple распределяет задачи по процессорам так, чтобы код масштабировался до количества доступных процессоров.

•Несколько алгоритмов, написанных с использованием модели программирования задач, могут выполняться одновременно без существенного влияния на производительность.

•Сложные проблемы могут быть решены без использования традиционных инструментов синхронизации, таких как мьютексы и переменные условия.

•Если такие средства синхронизации не используются, функция не может быть заблокирована.

•Функции из пакета Task просты, и модель отражает вызов обычных функций.

Таблица 1. Список команд пакета Task

Команда	Параметры	Описание команды
Start(fcn, arg1, ..., argN)	fcn – функция для выполнения в Task. args1..argsN – аргументы для fcn или спецификация дочерних задач.	Функция Start создает начальную задачу и ждет, пока задача не будет завершена, прежде чем вернуть значение, возвращенное этой

		задачей.
Continue(fcn, arg1, ..., argN)	fcn – функция для выполнения в Task. args1..argsN – аргументы для fcn или спецификация дочерних задач.	Функция Continue вызывается из Task для создания дочерних задач и задачи продолжения.
Return(value)	value – возвращаемое значение.	Функция Return позволяет задаче остановить выполнение модели задачи и вернуть определенное значение из Start.

Рассмотрим пример объявления функции в Maple (листинг 1).

Листинг 1. Пример объявления функции в Maple.

```
>   f := proc( args )
    # work
    ...
    return cont( fc1( c1args ), fc2( c2args ) );
end proc;
```

Чтобы вычислить f, мы выполняем некоторую работу: выполняем fc1, затем fc2, затем, наконец, выполняем cont, передавая возвращаемые значения fc1 и fc2 в качестве аргументов.

В модели программирования Task вместо выполнения fc1, fc2 и cont в одном потоке мы создаем задачу для каждой функции. Задача – это небольшая единица работы, представленная в Maple вызовом

функции, то есть процедурой и аргументами этой процедуры. Некоторые аргументы процедуры могут быть результатом действия других задач. Если все аргументы задачи доступны, то она может быть выполнена. Если некоторые аргументы недоступны, то задача ожидает получения результатов. В этом примере fc1 и fc2 могут быть выполнены немедленно, а cont – нет. Когда fc1 и fc2 завершены, их возвращаемые значения передаются в cont. Когда функция cont получает свои аргументы, её можно выполнить.

Когда задачи порождают новые задачи, создается дерево зависимостей, в котором одни задачи ждут завершения других задач, прежде чем они смогут выполняться.

Вот простой пример реализации сложения чисел заданного диапазона (листинг 2).

Листинг 2. Параллельная реализация программы сложения чисел

```
> cont := proc( a, b )
   return a + b;
end proc;
task := proc( i, j )
   local k;
   if ( j-i < 1000 ) then
       return add( k, k=i..j );
   else
```

```
        k := floor( (j-i)/2 )+i;

        Threads:-Task:-Continue( cont, Task=[ task,
i, k ], Task=[ task, k+1, j ] );

    end if;

end proc;

Threads:-Task:-Start( task, 1, 10^8 );
```

Start создает первоначальную задачу, а затем ждет, пока эта задача не будет завершена. Proc – определение тела процедуры (функции).

Функция Add используется для суммирования диапазона чисел. Другая функция Floor округляет значение параметра в меньшую сторону.

Конструкция Threads:-Task:-Start(task, 1, 10^8) создает задачу, которая выполняет функцию task(1,10^8). Объявленная в примере функция task проверяет размер диапазона, и если он достаточно мал, то add просто вычисляется напрямую. Если диапазон велик (включает 1000 или более чисел), то мы делим его пополам и создаем две дочерние задачи с помощью конструкции Threads:-Task:-Continue(cont, Task=[task, i, k], Task=[task, k+1, j]). Эти дочерние задачи выполняют task(i, k) и task(k+1, j). Функция Continue также создает третью задачу продолжения, роль которой играет функция cont. Задача продолжения – это задача, ожидающая возврата значений от двух дочерних задач. Она занимает место текущей задачи в дереве зависимостей. Это позволяет завершить текущую задачу, оставив дерево зависимостей

нетронутым. Задача продолжения может объединять результаты дочерних задач перед передачей их родительской задаче.

Для сравнения напишем последовательную версию программы сложения диапазона чисел (листинг 3).

Листинг 3. Последовательная реализация программы сложения чисел

```
Cont:=proc(b::posint)
        Local k,z;
for z from 0 by 1 to b do
            k:=k+z;
end do;
        return k;
end proc;
cont(10^7)
```

Для сравнения времени выполнения параллельной и последовательной версий программы было проведено по пять тестов для разных диапазонов чисел: $1..10^7$, $1..10^8$, $1..10^9$, $1..10^{10}$. Определено среднее время расчета каждого из этих диапазонов (рис. 1).

Последовательное выполнение программы				
	10^7	10^8	10^9	10^10
1 тест	9,1	89,12	907,24	11473,67
2 тест	9,25	87,77	878	11022,35
3 тест	8,75	88,07	891,92	11230
4 тест	9	89,36	901,34	11652,7
5 тест	9,53	87,53	880,9	11378,54
Среднее	9,126	88,37	891,88	11351,452
Параллельное выполнение программы				
	10^7	10^8	10^9	10^10
1 тест	3,98	26,74	300,6	4204,07
2 тест	3,15	29	316,25	4555,78
3 тест	2,78	33,57	289,61	4376,53
4 тест	3,35	28,8	304,57	4635,32
5 тест	2,97	28,14	305,75	4245,76
Среднее	3,246	29,25	303,356	4403,49

Рис. 1. Сравнительная таблица времени выполнения последовательной и параллельной версий программы

На основании данных из таблицы (рис. 1) была построена сравнительная диаграмма (рис. 2).

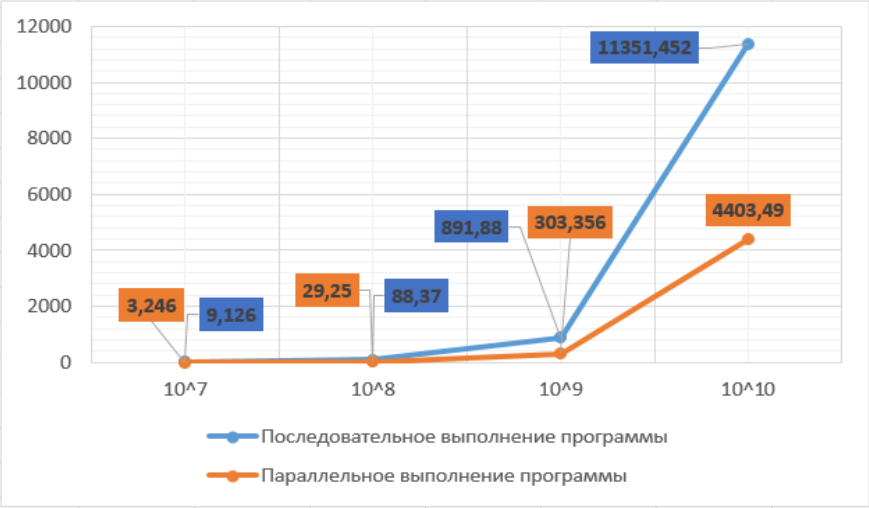

Рис. 2. Сравнительная диаграмма времени выполнения вычислений для последовательной и параллельной версии программы

Данная диаграмма (рис. 2) показывает, что для вычисления небольших значений можно использовать последовательную версию программы, так как, во время выполнения, она использовала меньшее количество памяти, в то время, как разница во времени была почти незначительной, однако, для работы с большими числами, лучше

использовать параллельную версию программы, для уменьшения времени вычислений. Вычисления проводились на персональном компьютере со следующими характеристиками: Процессор Intel Core i7-6700HQ CPU 2.60GHz, 4 ядра, 8 ГБ оперативной памяти. Программное обеспечение: ОС Windows 10, MAPLE 2022.

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 Programming // MapleSoft.ru. URL: https://www.maplesoft.com/support/help/Maple/view.aspx?path=ProgrammingGuide%2FChapter15.
2. Parallel Programming: The Task Programming Model // MapleSoft.ru. URL: https://www.maplesoft.com/support/help/Maple/view.aspx?path=updates/Maple16/ParallelProgramming.
3. Multithreaded Programming // MapleSoft.ru. URL: https://www.maplesoft.com/support/help/Maple/view.aspx?path=multithreaded.
4. Grid Computing (Parallel Distributed Computing) // MapleSoft.ru. URL:

https://www.maplesoft.com/support/help/Maple/view.aspx?path=updates%2FMaple16%2FGridComputing.

5. Overview of the Grid Computing Toolbox // MapleSoft.ru. URL: https://www.maplesoft.com/support/help/Maple/view.aspx?path=Grid%2FOverview