Лекция 5. Пул потоков

ИУ8

November 7, 2017

Мотивация

Объект потока

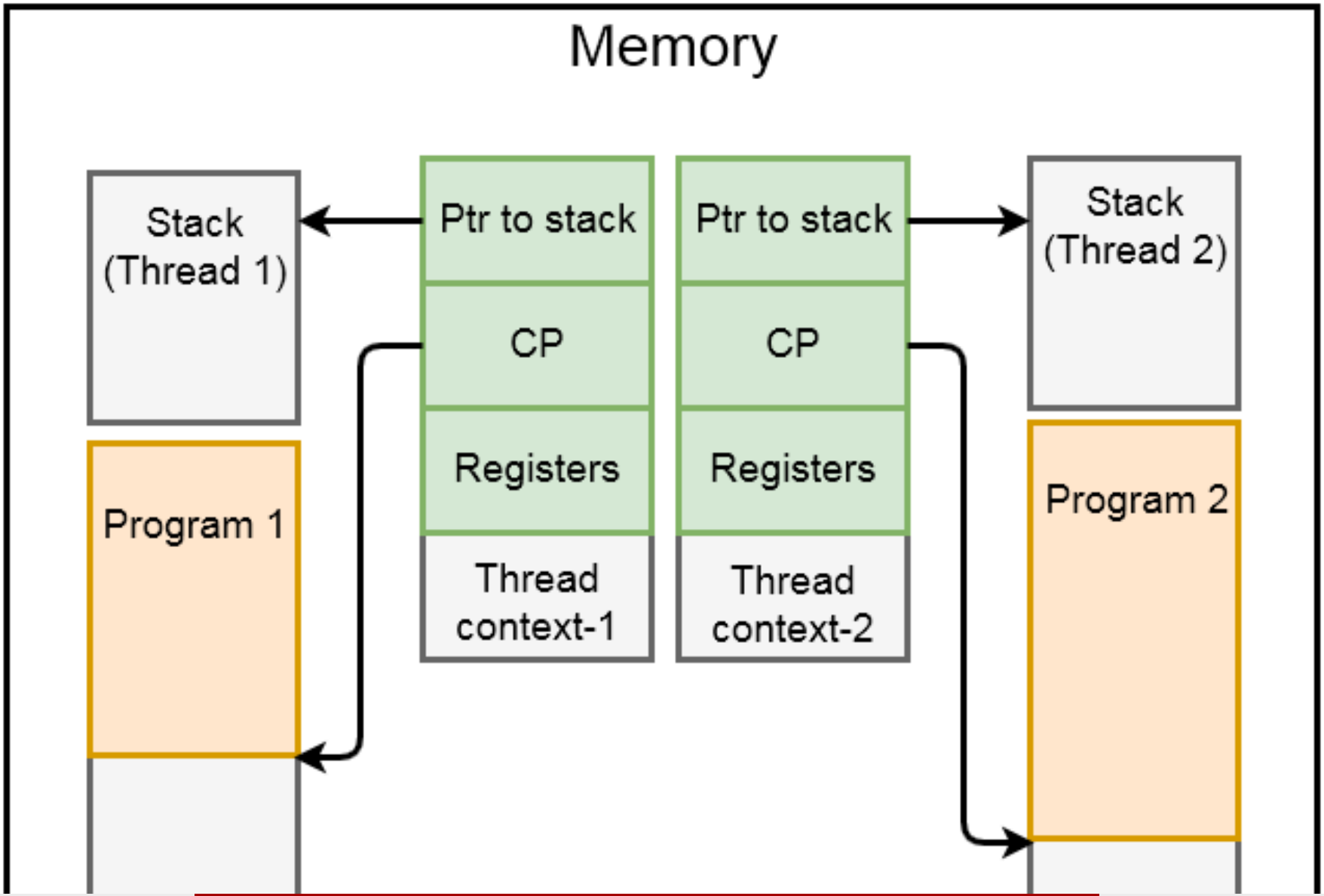
Чтобы ОС поддерживала многозадачность, каждый выполняемый поток должен обладать своим контекстом исполнения.

Этот контекст используется для хранения данных о текущем состоянии потока

Таким образом контекст состоит как минимум из

- значения регистров процессора
- указателя на стек потока
- счетчика команд

Мотивация


Memory
Stack (Thread 1)
Ptr to stack
CP
Registers
Thread context-1
Ptr to stack
CP
Registers
Thread context-2
Stack (Thread 2)
Program 1
Program 2

Мотивация

Таким образом, каждый новый поток занимает некоторый объем оперативной памяти.

Идем дальше.

Мотивация

Переключение контекста

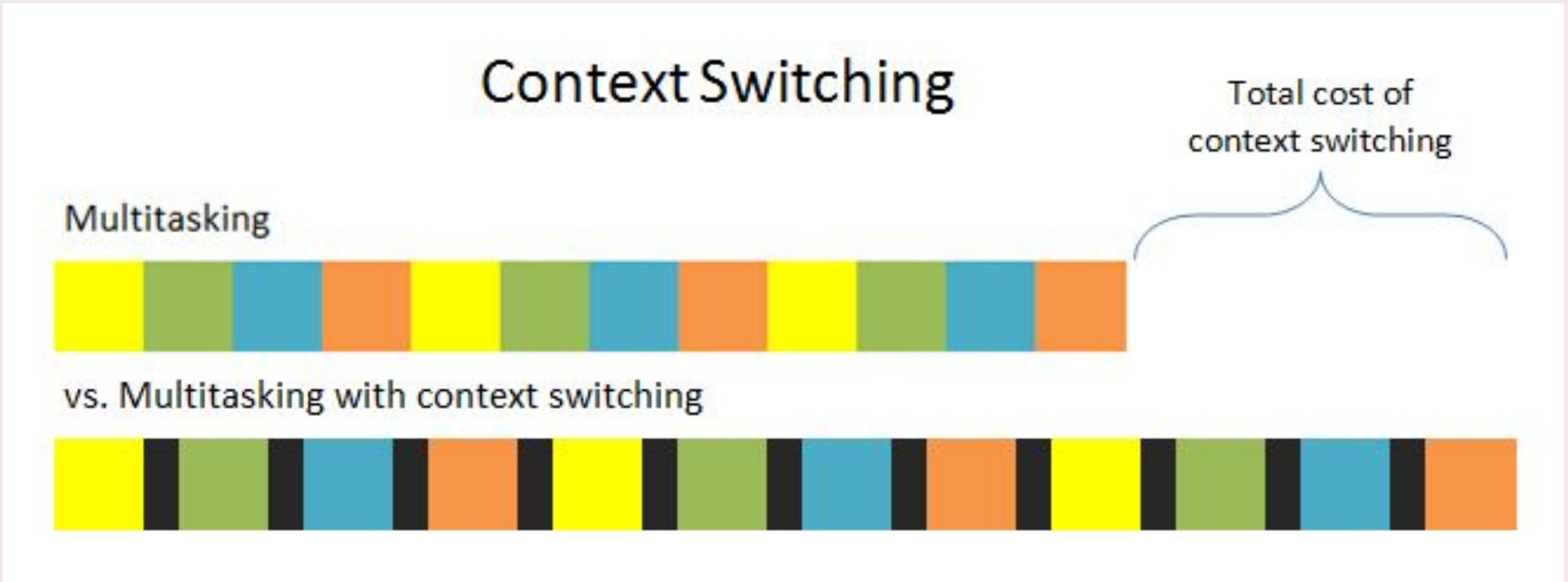
В системе количество потоков может превышать число ядер, тогда будет применяться механизм переключения задач.

В произвольный момент времени ОС передает поток на исполнение процессору. Этот поток исполняется в течение некоторого временного интервала. После завершения этого интервала контекст ОС переключается на другой поток. При этом происходит следующее:

1. обновляется контекст потока;
2. из набора имеющихся потоков выбирается один, который будет исполняться на процессоре;
3. значения из выбранного контекста потока загружаются во внешнее окружение.

Мотивация

Переключение контекста



Мотивация

Резюме

Получаем, что новые потоки не только занимают объем оперативной памяти, но и создают дополнительные временные издержки, так как операционной системе приходится планировать исполнение потоков и выполнять переключения контекста.

Пул потоков

Пул потоков

Во многих системах бессмысленно заводить отдельный поток для каждой задачи, которая потенциально может выполняться одновременно с другими. Тем не менее, хотелось бы пользоваться преимуществами параллелизма там, где это возможно.

Пул потоков предоставляет именно такие возможности: задачи, которые могут выполняться параллельно, отправляются в пул, а тот ставит их в очередь ожидающих работ. Затем один из рабочих потоков забирает задачу из очереди, исполняет ее и принимается за следующую задачу.

Пул потоков

Простейший пул потоков

```
1  class thread_pool {
2    std::atomic_bool done = false;
3    thread_safe_queue<std::function<void()>> work_queue;
4    std::vector<std::thread> threads;
5
6    void worker_thread(){
7      while(!done.load()) {
8        std::function<void()> task;
9        if(work_queue.try_pop(task)) {
10         task();
11       } else {
12         std::this_thread::yield();
13       }
14     }
15   }
16 public :
17   ~thread_pool() {
18     done.store(true);
19   }
```

Пул потоков

Простейший пул потоков: продолжение

```
1   thread_pool(){
2     const int thread_count = std::thread::hardware_concurrency();
3     try {
4       for(auto i = 0; i < thread_count; ++i) {
5         threads.push_back(std::thread(
6           &thread_pool::worker_thread,
7           this));
8       }
9     } catch(...) {
10      done.store(true);
11    }
12  }
13
14  template<typename Func>
15  void submit(Func f) {
16    work_queue.push(std::function<void()>(f));
17  }
18 }; // end of thread_pool
```

Пул потоков

Пул потоков с возможностью получения результата задачи

см thread_pool.h

Пул потоков

Как модифицировать пул

- динамически изменять количество рабочих потоков
- для каждого потока создать свою очередь задач
- предоставить доступ к списку задач
- добавить в пул потоков дополнительных рабочих

Пул потоков

Динамическое число рабочих

Суть: в зависимости от количества задач изменять количество рабочих потоков

- \+ не занимаем ОП под потоки, которые ожидают задач
- \+ снижаем конкуренцию при доступе к очереди задач
- \+ уменьшаем количество потоков в ОС
- \- усложняется логика работы пула
- \- тратим дополнительное время на создание потока

Пул потоков

Отдельные очереди задач

Суть: чтобы общая очередь задач не стала узким местом, создаем по очереди задачи для каждого потока

- \+ сводим конкуренцию за доступом к очереди задач к минимуму
- \- усложняется логика работы пула
- \- возможна ситуация обработки наиболее «длительных» задач только одним потоком, тогда как остальные потоки будут простаивать

Пул потоков

Доступ к ресурсам пула

Суть: если мы имеет дополнительного вычислительного ресурса, то можем выполнять задачи из очереди задач пула потоков

\+ динамически увеличиваем вычислительные способности пула

\+ уменьшаем вероятность блокировки задач

\- усложнение логики работы пула

The end

Самостоятельное изучение

- Реализуйте модификации пулов потоков, описанные
- boost::thread::interrupt

Список литературы

- Уильямс - Параллельное программирование на C++ в действии
- http://www.jlc.tcu.edu.tw/OS/932/threadmanager.pdf
- https://www.cs.uic.edu/ jbell/CourseNotes/OperatingSystems/4_Threads.html
- https://habrahabr.ru/post/306332/