Операционные системы

Межпроцессное взаимодействие

Межпроцессное взаимодействие

Реализация блокировок и синхронизация потоков в OpenMP

Блокировки (замки) (1)

- OpenMP включает набор функций, предназначенные для синхронизации кода с использованием блокировок.
- OpenMP два типа блокировок:
  - простые блокировки;
  - рекурсивные (nestable) блокировки.
- Блокировки обоих типов могут находиться в одном из трех состояний:
  - неинициализированном;
  - заблокированном;
  - разблокированном.

Блокировки (замки) (2)

- Простые блокировки (omp_lock_t) не могут быть установлены более одного раза, даже тем же потоком.
- Рекурсивные блокировки (omp_nest_lock_t) идентичны простым с тем исключением, что, когда поток пытается установить уже принадлежащую ему рекурсивную блокировку, он не блокируется. Кроме того, OpenMP ведет учет ссылок на рекурсивные блокировки и следит за тем, сколько раз они были установлены.
- OpenMP предоставляет подпрограммы, выполняющие операции над этими блокировками. Каждая такая функция имеет два варианта: для простых и для рекурсивных блокировок.

Функции для работы с блокировками в OpenMP и Win32

- Вы можете выполнить над блокировкой пять действий: инициализировать ее, уничтожить , установить (захватить), освободить, проверить.

Простая блокировка OpenMP	Рекурсивная блокировка OpenMP	Win32-функция
omp_lock_t	omp_nest_lock_t	CRITICAL_SECTION
omp_init_lock	omp_init_nest_lock	InitializeCriticalSection
omp_destroy_lock	omp_destroy_nest_lock	DeleteCriticalSection
omp_set_lock	omp_set_nest_lock	EnterCriticalSection
omp_unset_lock	omp_unset_nest_lock	LeaveCriticalSection
omp_test_lock	omp_test_nest_lock	TryEnterCriticalSection

Пример использования блокировок

```
omp_lock_t lck;
omp_init_lock(&lck);
…
omp_set_lock(&lck);
…
omp_unset_lock(&lck);
```

Барьерная синхронизация

- При одновременном выполнении нескольких потоков часто возникает необходимость их синхронизации. OpenMP поддерживает несколько типов синхронизации, помогающих во многих ситуациях.
- Один из типов - неявная барьерная синхронизация, которая выполняется в конце каждого параллельного региона для всех сопоставленных с ним потоков. Механизм барьерной синхронизации таков, что, пока все потоки не достигнут конца параллельного региона, ни один поток не сможет перейти его границу.

Неявная барьерная синхронизация

- Неявная барьерная синхронизация выполняется также в конце каждого блока #pragma omp for, #pragma omp single и #pragma omp sections.
- Чтобы отключить неявную барьерную синхронизацию в каком-либо из этих трех блоков разделения работы, укажите раздел nowait.

```
#pragma omp parallel
{
    #pragma omp for nowait
    for(int i = 1; i < size; ++i)
        x[i] = (y[i-1] + y[i+1])/2;
}
```

- Синхронизация потоков в конце цикла for не будет выполняться, хотя в конце параллельного региона они все же будут синхронизированы.

Типы явной синхронизации

- atomic
- critical
- barrier
- master
- ordered
- flush

Атомарные операции

- Директива atomic может быть применена только для простых выражений, но является наиболее эффективным средством организации взаимоисключения.
- Позволяет выполнить операцию в атомарном режиме (неделимая операция). В этом случае происходит предотвращение прерывания доступа, чтения и записи данных, находящихся в общей памяти, со стороны других потоков.

```
#pragma omp atomic
<операторы программы>
```

Критические секции

- Для создания критических секций в OpenMP применяется директива #pragma omp critical [имя]. Она имеет такую же семантику, что и критическая секция Win32.
- Вы можете использовать именованную критическую секцию, и тогда доступ к блоку кода является взаимоисключающим только для других критических секций с тем же именем (это справедливо для всего процесса).
- Если имя не указано, директива ставится в соответствие некоему имени, выбираемому системой. Доступ ко всем неименованным критическим секциям является взаимоисключающим.

```
#pragma omp critical [ name ]
<структурный блок программы>
```

Явная барьерная синхронизация

- Для включения в код явной барьерной синхронизации используйте директиву barrier.

```
#pragma omp barrier
```

Директива master

- В ряде случаев требуется, чтобы блок кода был выполнен основным потоком.
- В этом случае применяется директива #pragma omp master. В отличие от директивы single при входе в блок master и выходе из него нет никакого неявного барьера.

Упорядочивание итерация с помощью ordered

- Директивы ordered определяют блок внутри тела цикла, который должен выполняться в том порядке, в котором итерации идут в последовательном цикле.

```
#pragma omp ordered
<структурный блок>
```

Применение ordered

- Неправильно:

```
#pragma omp parallel for
   ordered
for (int i = 0; i <10; i++)
{
   myFunc(i);
}
```

- Правильно:

```
#pragma omp parallel for
   ordered
for (int i = 0; i <10; i++)
{
   #pragma omp ordered
   {
     myFunc(i);
   }
}
```

Явный барьер памяти

- В OpenMP реализована слабая модель памяти.
- Директива flush позволяет определить точку синхронизации, в которой системой должно быть обеспечено единое для всех потоков состояние памяти (т.е. если потоком какое-либо значение извлекалось из памяти для модификации, измененное значение обязательно должно быть записано в общую память).

  #pragma omp flush [(list)]

- Директива содержит список list с перечнем переменных, для которых выполняется синхронизация. При отсутствии списка синхронизация выполняется для всех переменных потока.

Неявный барьер памяти

- В директиве barrier.
- При входе и выходе из параллельной секции директив parallel, critical, ordered.
- При выходе из параллельной секции директив for, sections, single.
- При входе и выходе из параллельной секции директив parallel for, parallel sections.

Отказ от неявного барьера памяти

- При входе в параллельную секцию директивы for.
- При входе и выходе из секции директивы master.
- При входе в параллельную секцию директивы sections.
- При входе в секцию директивы single.
- При выходе из секции директив for, single или sections, если к директиве применено выражение nowait.