Знакомство с технологией OpenMP

Марчевский И.К., Попов А.Ю.

МГТУ им. Н.Э. Баумана

Стандарт OpenMP I

OpenMP (Open Multi-Processing) — программный интерфейс (API) для разработки параллельных программ на языках C/C++ и Fortran для систем с общей памятью.

- Позволяет производить распараллеливание на высоком уровне — путем добавления в коде директив компилятора, в первую очередь — для циклов. Не требуется вручную разделять данные и управлять параллельными процессами вычислений и передачи данных — это делается компилятором;
- фактически используемые в OpenMP элементы для распараллеливания:
  - директивы компилятора;
  - runtime-функции библиотеки OpenMP;
  - переменные среды;

Стандарт OpenMP II

- при запуске в последовательном режиме директивы просто игнорируются (как комментарии), что позволяет иметь одну программу для обоих режимов — удобно для тестирования и отладки;
- распараллеливание чаще всего производится поверх имеющейся последовательной программы, в отличие от технологий MPI и CUDA, которые почти всегда требуют учета архитектурных особенностей на стадии реализации алгоритмов. Более того, распараллеливание часто выполняется инкрементально — путем перехода к параллельному исполнению наиболее трудоемких участков из оставшихся;
- OpenMP — один из наиболее простых для программиста способов написания параллельных программ для многоядерных систем с общей памятью (аналог — Pthreads, которая лежит в основе OpenMP и является низкоуровневой библиотекой).

История разработки и развития OpenMP

Версии спецификации OpenMP:

- 1.0 (Fortran) — октябрь 1997;
- 1.0 (C/C++) — октября 1998;
- 1.1 (Fortran) — ноябрь 1999;
- 2.0 (Fortran) — ноябрь 2000, 2.0 (C/C++) — март 2002;
- 2.5 (единая) — май 2005 (параллелизм на уровне не только циклов, но и задач);
- 3.0 (C/C++) — 2008, 3.0 (Fortran) — 2009 (поддержка объекта task общего вида);
- 4.0 — 2013, 4.5 — 2015 (поддержка SIMD, пользовательских редукций);
- 5.0 — ноябрь 2018, 5.1 — ноябрь 2020; 5.2 — ноябрь 2021;
- 6.0 (preview) — ноябрь 2022.

Поддержкой и развитием стандарта OpenMP занимается некоммерческая организация OpenMP ARB (Architecture Review Board).

Поддержка OpenMP компиляторами I

OpenMP поддерживается основными компиляторами C/C++, но в различной спецификации.

GNU C Compiler (GCC)

GCC 4.2 (05.2007) — OpenMP 2.5; GCC 4.4 (2009–2012) — OpenMP 3.0;
GCC 4.9 (2014–2016) — OpenMP 4.0;
GCC 11-14 (текущая версия, OpenMP 5.0-5.2 в различном объеме).
Использование: `gcc -o binaryname -fopenmp sourcename.c`

Microsoft Visual Studio

Поддержка OpenMP 2.0
Использование: Свойства проекта — «Свойства конфигурации»/ «C/C++»/«Язык» — «Поддержка OpenMP» — «Да».

Частичная поддержка 3.0, использование:
Свойства проекта — «Свойства конфигурации»/«C/C++»/«Язык» — «Включить эксперментальные модули поддержки C++» — «Да».

Поддержка OpenMP компиляторами II

Intel C++ Compiler

Версия 2021.1 — OpenMP 4.5 и частично 5.0;
версия 2022.3 — OpenMP 4.5 и частично 5.1.
Использование:
Visual Studio: Свойства проекта — «Свойства конфигурации»/ «C/C++»/«Language (Intel C++)» — «OpenMP Support» — «Generate Parallel Code»,
Linux — ключ `-qopenmp`

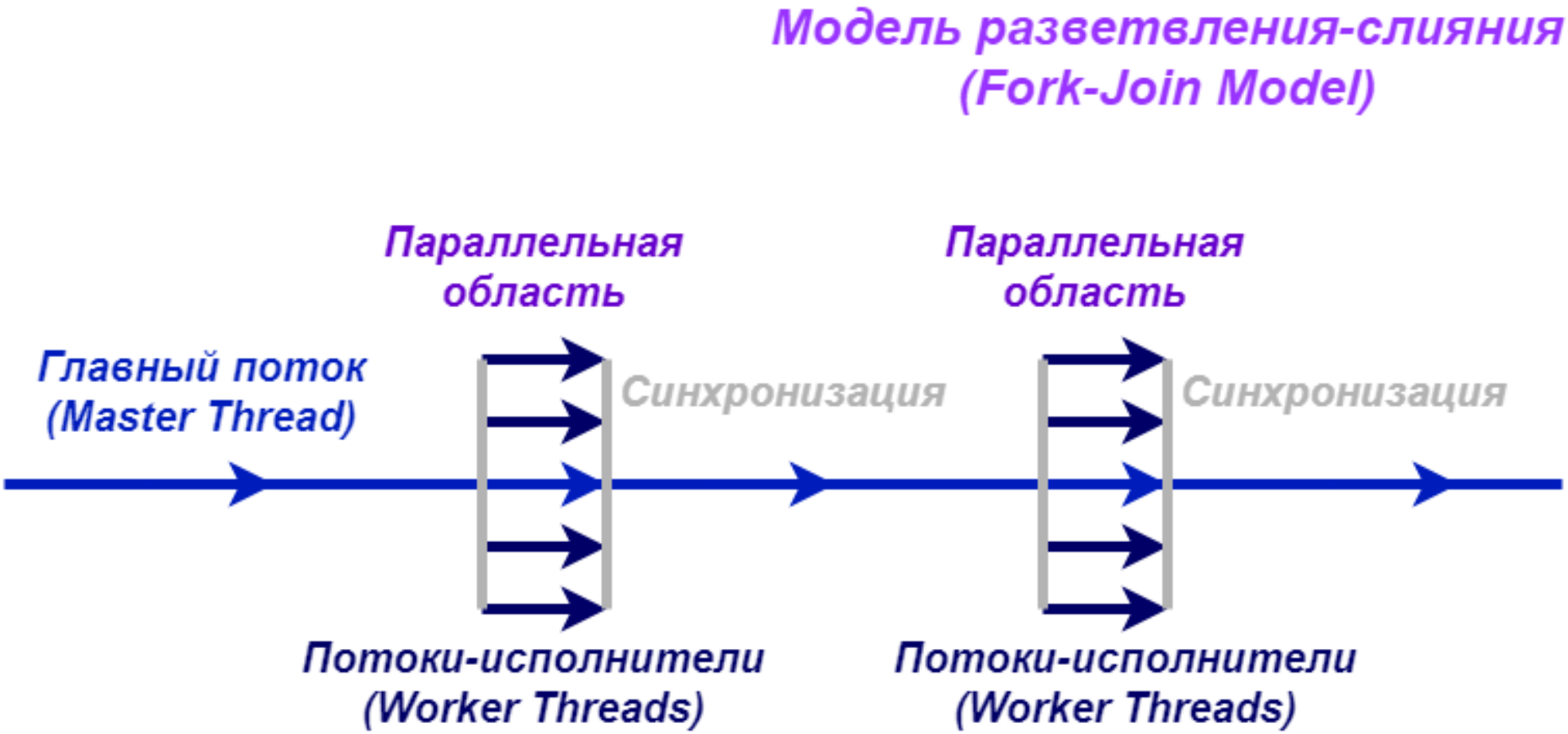
Clang

Версия 3.9 — OpenMP 4.5; версия 11 — OpenMP 5.0.

ARM

Поддержка OpenMP 3.1 и частично 4.0/5.0/5.1
Использование — ключ `-fopenmp`

Модель параллелизма в OpenMP



Директивы в OpenMP

Директивы OpenMP указываются с помощью
`#pragma omp <директива> <опции (clause)>`

Обозначение параллельной секции (фигурные скобки могут отсутствовать при наличии всего 1 внешнего блока):

```
#pragma omp parallel
{
    код
}
```

Директивы в OpenMP

Директивы OpenMP указываются с помощью
`#pragma omp <директива> <опции (clause)>`

Обозначение параллельной секции (фигурные скобки могут отсутствовать при наличии всего 1 внешнего блока):

```
#pragma omp parallel
{
    код
}
```

Задание кол-ва нитей:

- переменные среды `OMP_NUM_THREADS` и `NUMBER_OF_PROCESSORS`, в т.ч. в рамках конкретной сессии;
- опции и функции OpenMP.

Стек механизмов, используемых в OpenMP

Управление числом нитей (потоков) в OpenMP

- Опция `num_threads(n)` директивы `parallel` (имеет высший приоритет как более близкая к коду);
- функция `omp_set_num_threads(n)`. Требует заголовочного файла `omp.h`

Получение информации о параллельных нитях:

- `omp_get_num_threads()` — фактическое число нитей в параллельной секции;
- `omp_max_num_threads()` — число нитей, которое может быть порождено в параллельной секции;
- `omp_get_num_thread()` — id текущего потока.

Вложенный параллелизм

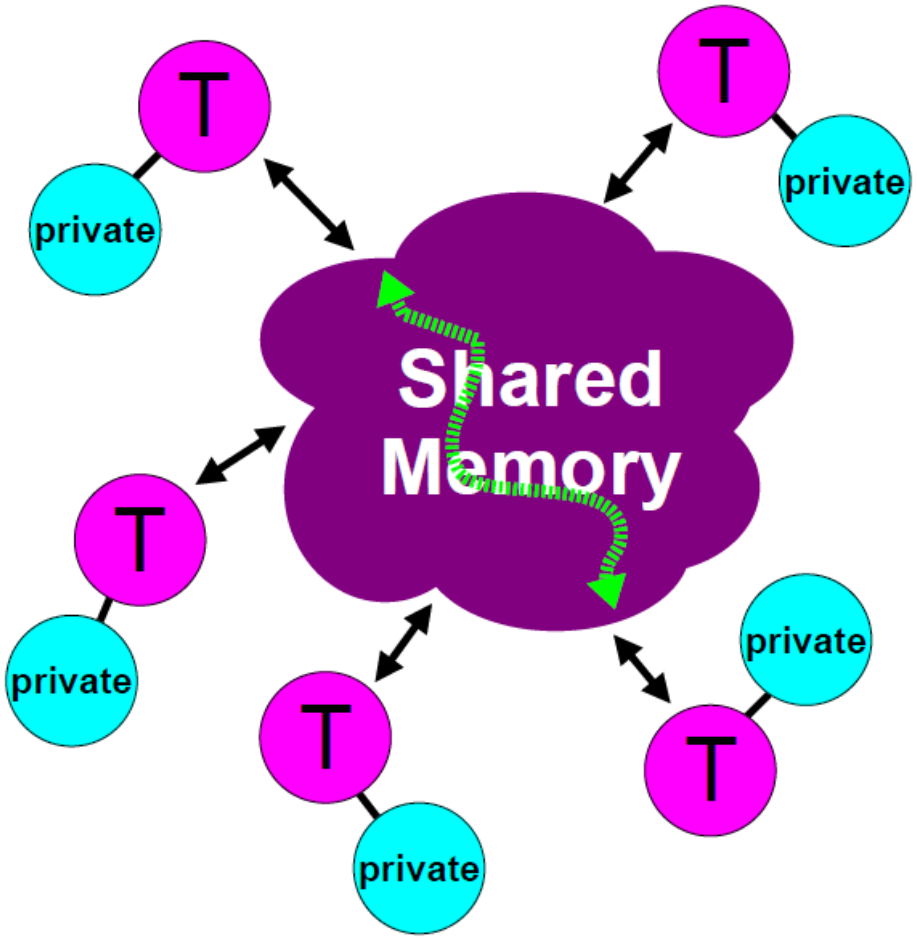
Порождение параллельных секций

- происходит динамически, в процессе работы программы (не определяются заранее компилятором);
- затратная операция, следует ограничивать их количество и включать в них (разумно) больше кода;
- можно использовать опцию `if(условие)` директивы `parallel`, которая ограничит порождение только при выполнении условия (в противном случае — последовательное выполнение).

Параллельные секции могут вложены друг в друга (по умолчанию такая возможность отключена). Включение:

- функция `omp_set_nested(1)` (1 — произвольное число больше 0);
- функция `omp_set_max_active_levels(число уровней)` — на компиляторах Intel (вложенный параллелизм — 2 и более уровней)

Модель работы с памятью в OpenMP



- По умолчанию переменные программы являются общими для всех потоков внутри параллельной секции;
- указать необходимость создания собственного экземпляра переменной у каждого потока можно с помощью опции `private(переменная)`

Управление переменными в параллельных секциях кода

```
#pragma omp parallel shared(a,b) private(c,d) \
firstprivate(e,f) default(none)
```

1. `shared` — общие для всех нитей переменные;
2. `private` — переменные, для которых в каждой нити создается свой экземпляр;
3. `firstprivate` — аналогично `private`, но переменные инициализируются значениями из главного потока;
4. `default` — тип видимости переменных по умолчанию — `shared` либо `none`.

При этом видимость констант не нужно указывать явно (изначально видны всем нитям), при использовании `private` (`firstprivate`) после выхода из параллельной секции переменная в главном потоке имеет то же значение, что и до нее.