

OpenMP + MPI

Эдуард Храмченков

Гибридная параллелизация

- Многие современные кластеры гибридные суперкомпьютеры
- В каждом из узлов установлены многоядерные процессоры
- При использовании чистого MPI многоядерность процессора на узлах используются неоптимальным образом
- Можно совместить использование OpenMP и MPI

Гибридная параллелизация

- Две степени параллелизации кода «грубая» и «тонкая»
- «Грубая» параллелизация разделение кода/данных на большие блоки
- «Тонкая» паралллелизация на уровне отдельных циклов
- Для «грубой» параллелизации используется механизм MPI, для «тонкой» - OpenMP

Гибридная параллелизация

- Преимущества гибридной схемы
 - Лучшая сбалансированность вычислений
 - Уменьшение передачи данных
 - Автоматическая когерентность данных в пределах узла
- Недостатки
 - Дополнительные затраты на переработку алгоритма – неоптимизированный алгоритм может работать медленнее

Возможные конфигурации

- Каждый узел как SMP
 - Один MPI процесс на узел
 - Параллельные потоки делят между собой всю память узла
 - Количество потоков зависит от конфигурации узла
- Каждый процессорный сокет как SMP
 - Один MPI процесс на сокет
 - Параллельные потоки делят память конкретного сокета
 - Количество потоков = количество ядер процессора в сокете

Базовая схема кода

```
MPI_Init(...)
MPI_Func(...)
#pragma omp parallel
       [MPI_Func(...)]
MPI_Func(...)
MPI_Finalize()
```

Базовая схема кода

- Последовательный код исполняется мастер потоком – соответствующим процессом MPI
- Группа OpenMP потоков создается внутри процесса
- Каждая группа потоков знает номер своего процесса
- Вызов МРІ внутри параллельного региона осуществляется согласно функции инициализации

Инициализация

 Чтобы процесс MPI мог работать в многопоточном режиме необходима специальная инициализация окружения MPI

```
MPI_Init_thread(
    int *argc,
    char **argv,
    int required,
    int *provided)
```

 Функция замещает стандартную функцию инициализии MPI_Init



Инициализация

- Аргумент required содержит желаемый уровень многопоточности
- Аргумент provided содержит уровень многопоточности, предоставляемый реализацией MPI
- Сравнение аргументов способ определения корректности работы кода с потоками
- Значения аргументов приведены по возрастающей

Инициализация

Уровень поддержки	Описание
MPI_THREAD_SINGLE	Исполняется только один поток
MPI_THREAD_FUNNELED	Многопоточный режим, только мастер-поток может вызывать функции MPI (по умолчанию)
MPI_THREAD_SERIALIZE	Многопоточный режим, любой поток может вызывать функции MPI, но не одновременно
MPI_THREAD_MULTIPLE	Многопоточный режим без ограничений



MPI_THREAD_FUNNELED

- Это минимальный уровень поддержки для многопоточных МРІ приложений
- Программист должен запрашивать минимально возможный уровень многопоточности
- При данном уровне многопоточности вызов MPI из многопоточного региона должен предваряться и завершаться директивой #pragma omp barrier

MPI_THREAD_FUNNELED

```
#pragma omp parallel
#pragma omp barrier
#pragma omp master
             MPI_Func(...);
#pragma omp barrier
```

MPI_THREAD_SERIALIZE

```
#pragma omp parallel
#pragma omp barrier
#pragma omp single
            MPI_Func(...);
//барьер не нужен – встроен в директиву single
```

Потокобезопасность

- Барьерная синхронизация при вызовах МРІ необходима для гарантии целостности состояния программы
- Если вызовы MPI осуществляет один поток все остальные бездействуют
- ▶ При использовании уровня МРІ_ТНЯЕАD_MULTIPLE следует быть осторожным – можно получить неопределенное состояние, deadlock, etc.

Пример 1

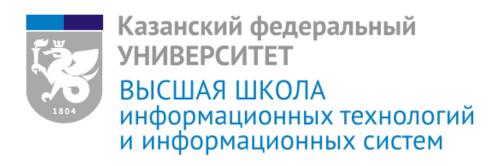
 Каждый МРІ процесс порождает несколько потоков

Задание

 Создать гибридный код вычисляющий число π по формулам

$$\pi = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$$

$$f(x) = \frac{4}{1+x^2}; x_i = \frac{i-0.5}{n}$$



Вопросы

ekhramch@kpfu.ru