Title

Курс «Параллельное программирование»

Евгений Юлюгин
yulyugin@gmail.com

15 марта 2014 г.

На прошлой лекции

- Основы MPI.

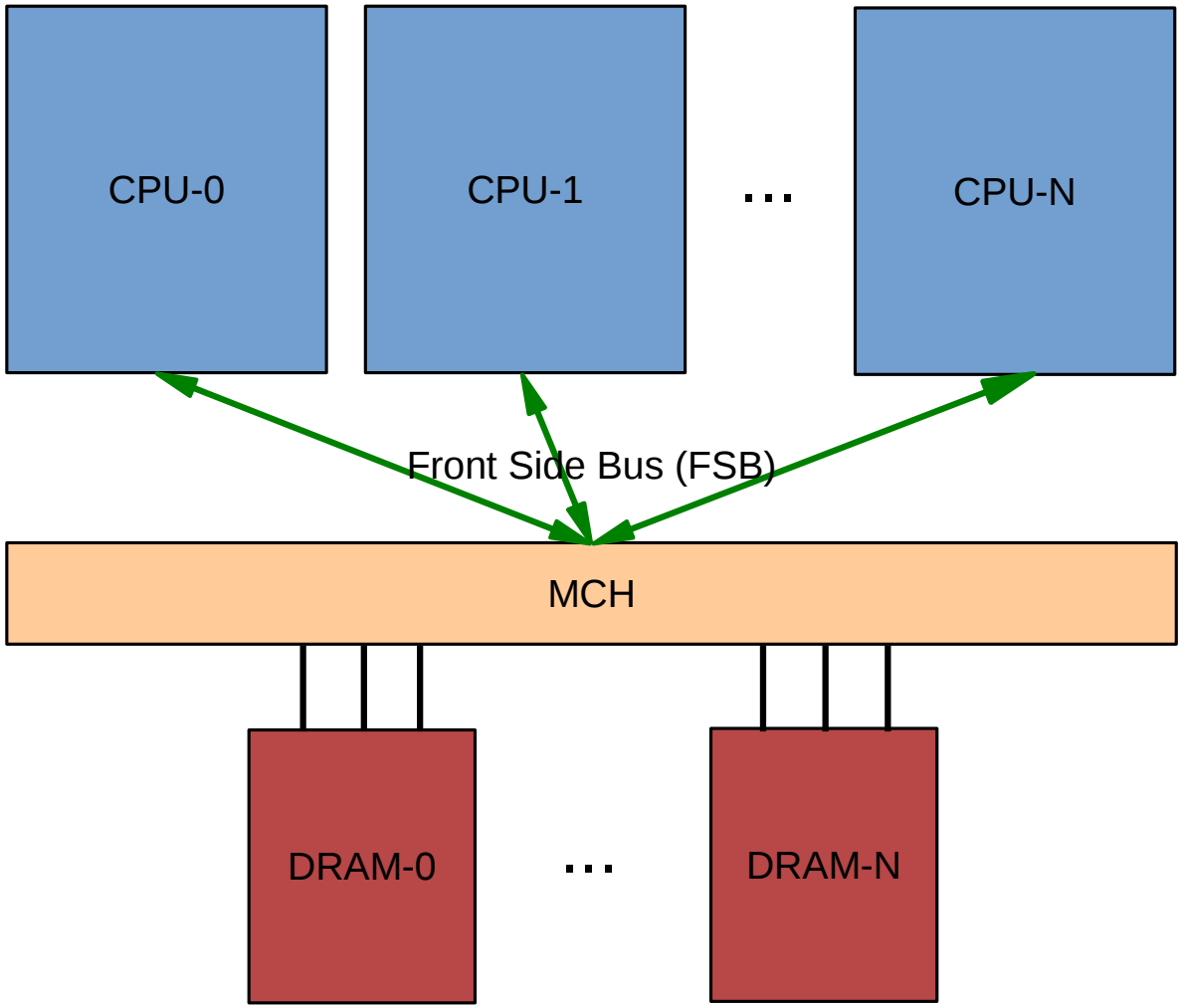
Классификация Флинна

	Single data	Multiple data
Single instruction	SISD	SIMD
Multiple instruction	MISD	MIMD

Симметричная мультипроцессорность

Симметричная мультипроцессорность (*англ.* Symmetric Multiprocessing, MPP) — архитектура вычислительных систем, в которой все процессоры подключаются к общей памяти (при помощи шины или подобного устройства) симметрично и имеют к ней однородный доступ.
Так же известна как UMA (Uniform Memory Access или Uniform Memory Architecture).

Симметричная мультипроцессорность



Массово-параллельная архитектура

Массово-параллельная архитектура (*англ.* Massive parallel processing, MPP) — класс архитектур, в которых процессоры имеют доступ исключительно к локальным ресурсам. То есть память разделена физически.
Не обеспечивает встроенного механизма обмена данными между узлами. Реализовывать коммуникации и распределение должен выполнять софт.

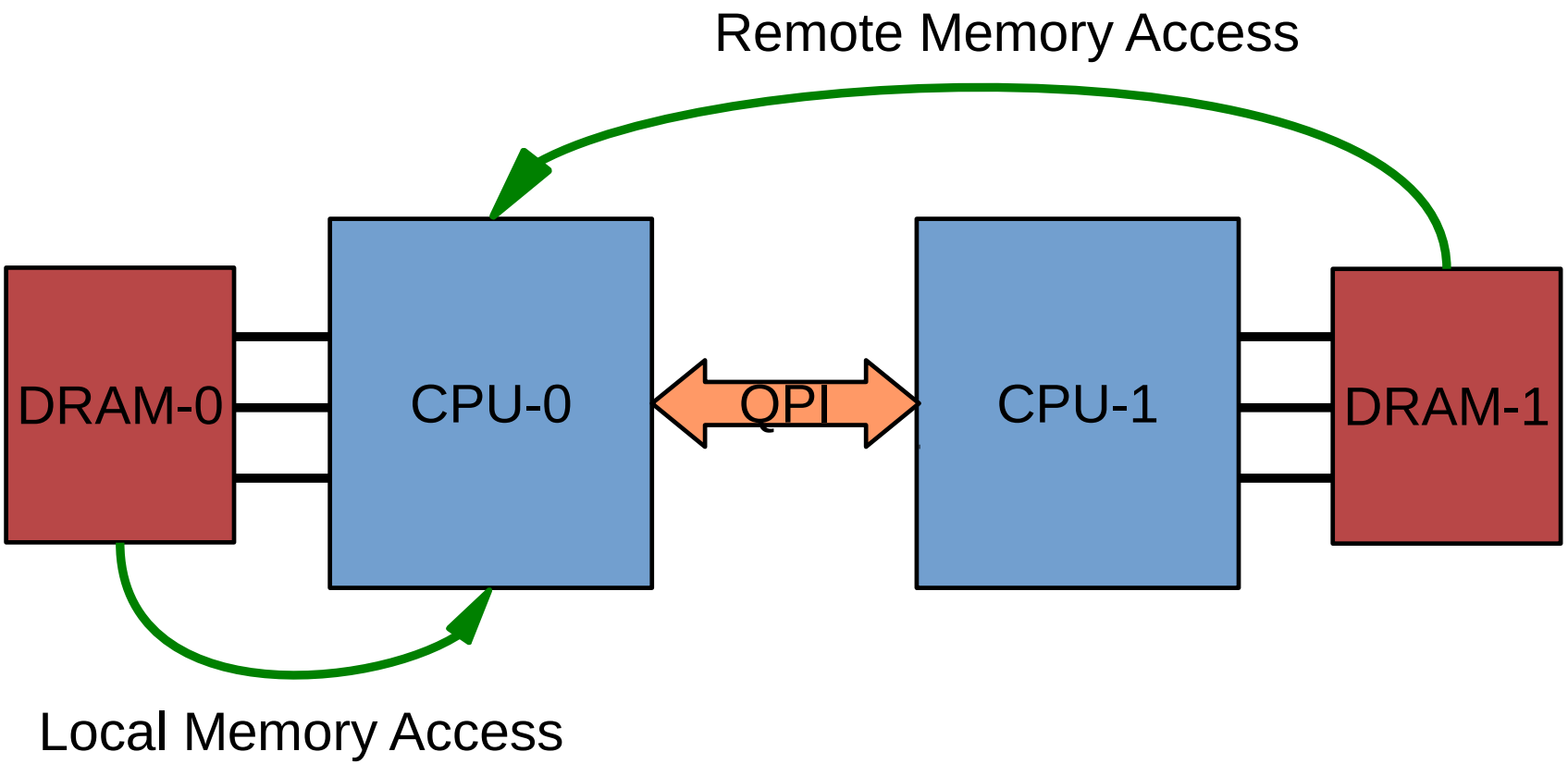
Массово-параллельная архитектура

TODO Add a picture.

Архитектура с неравномерной памятью

NUMA (Non-Uniform Memory Access или Non-Uniform Memory Architecture) система разделяется на множественные узлы, имеющие доступ как к своей локальной памяти, так и к памяти других узлов.

Архитектура с неравномерной памятью



Определение

Состояние гонки (*англ.* Race condition) — ошибка в многопоточной программе, при которой работа приложения зависит от того, в каком порядке выполняются части кода.
Свое название получила от похожей ошибки проектирования электронных схем (Гонки сигналов).
Состояние гонки — ошибка проявляющаяся в случайный момент времени.

Пример

```
int N = 1000;
int x = 0;
```

```
// thread 0
for (i = 0; i < N; ++i) {
    ++x;
}
```

```
// thread 1
for (i = 0; i < N; ++i) {
    if (x%2 == 0)
        printf("%d\n", x%2);
}
```

Семафор

Семафор — объект, ограничивающий количество потоков, которые могут войти в заданный участок кода.
Интерфейс семафора:

- init(n) — установить счетчик в n,
- enter() — подождать пока счетчик не станет больше нуля, затем уменьшить его на единицу,
- leave() — увеличить счетчик на еденицу.

Мьютекс

Мьютекс (*англ.* mutex) — «одноместный» семафор, служащий для синхранизации одновременно выполняющихся потоков.
TODO Mutex types.

Пример

```
int N = 1000;
int x = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

```
// thread 0
for (i = 0; i < N; ++i) {
    pthread_mutex_lock(&mutex);
    ++x;
    pthread_mutex_unlock(&mutex);
}
```

```
// thread 1
for (i = 0; i < N; ++i) {
    pthread_mutex_lock(&mutex);
    if (x%2 == 0)
        printf("%d\n", x%2);
    pthread_mutex_unlock(&mutex);
}
```

Взаимная блокировка

Взаимная блокировка (*англ.* deadlock) — ситуация в многозадачной среде, при которой несколько процессов находятся в состоянии бесконечного ожидания ресурсов, занятых самими этими процессами.

```
pthread_mutex_t A, B;
pthread_mutex_init(&A, NULL);
pthread_mutex_init(&B, NULL);
```

```
// thread 0
pthread_mutex_lock(&A);
pthread_mutex_lock(&B);
// ...
```

```
// thread 1
pthread_mutex_lock(&B);
pthread_mutex_lock(&A);
// ...
```

Livelock

Ситуация, когда в отличие от обычной блокировки процессы не зависают, а занимаются бесполезной работой.
Состояние системы постоянно меняется, но при этом она «зациклилась» и не производит полезной работы.

Алгоритм Деккера

```
bool flag[2] = {false, false};
bool turn = false; // or true
```

```
// thread 0
flag[0] = true;
while (flag[1]) {
    if (turn) {
        flag[0] = false;
        while (turn);
        flag[0] = true;
    }
}

// critical section
//...
turn = true;
flag[0] = false;
// end of critical section
// ...
```

```
// thread 1
flag[1] = true;
while (flag[0]) {
    if (!turn) {
        flag[1] = false;
        while (!turn);
        flag[1] = true;
    }
}

// critical section
//...
turn = false;
flag[0] = false;
// end of critical section
// ...
```

Задания

Вычислить $\sum_{n=0}^{N} \frac{1}{N!}$.

N — аргумент командной строки.

Построить график времени работы в зависимости от количества процессов и ускорения в зависимости от количества процессов.

На следующей лекции

Спасибо за внимание!

Замечание: все торговые марки и логотипы, использованные в данном материале, являются собственностью их владельцев. Представленная здесь точка зрения отражает личное мнение автора, не выступающего от лица какой-либо организации.