Средства и системы параллельного программирования

Семинар #4
Основы векторизации

Polus

http://hpc.cmc.msu.ru/polus - описание системы Polus

bjobs (-u all) посмотреть статус своих (всех) задач

ставить задания в очередь через планировщик bsub

Развёртка циклов

```
int i;
for ( i = 1; i < n; i++)
{
    a[i] = (i % b[i]);
}
```

```
int i;
for (i = 1; i < n - 3; i += 4)
{
    a[i] = (i % b[i]);
    a[i + 1] = ((i + 1) % b[i + 1]);
    a[i + 2] = ((i + 2) % b[i + 2]);
    a[i + 3] = ((i + 3) % b[i + 3]);
}
```

Развёртка циклов

```
int i;
for ( i = 1; i < n; i++)
{
    a[i] = (i % b[i]);
}
```

```
int i;
for (i = 1; i < n - 3; i += 4)
{
    a[i] = (i % b[i]);
    a[i + 1] = ((i + 1) % b[i + 1]);
    a[i + 2] = ((i + 2) % b[i + 2]);
    a[i + 3] = ((i + 3) % b[i + 3]);
}
```

Для чего мы делаем развертку циклов:

- убираем лишние операции (на проверку границ цикла)
- убираем зависимости между последовательными операциями

Выравнивание полей в C

`aligned (alignment)`

This attribute specifies a minimum alignment (in bytes) for variables of the specified type. For example, the declarations:

```
struct S { short f[3]; } __attribute__ ((aligned (8)));

typedef int more_aligned int  __attribute__ ((aligned (8)));
```

Для чего нужно выравнивание?

Выделение выровненного адреса

```
{
    void *mem = malloc(1024+16);
    void *ptr = ((char *)mem+16) & ~ 0x0F;
       // Use aligned pointer
    free(mem);
}
```

```
int posix_memalign(void **memptr, size_t alignment, size_t size);
```

```
void *aligned_alloc(size_t alignment, size_t size);
```

Объявление векторного типа

`vector_size (`*bytes*`)`

This attribute specifies the vector size for the variable, measured in bytes. For example, the declaration:

```
int foo __attribute__ ((vector_size (16)));
```

Интринсики SSE, AVX

Документация по интринсикам SSE и AVX

ARM NEON

Размер регистра SIMD = 128 байт

Документация по интринсикам ARM NEON

Задание

С помощью интринсик используемой целевой архитектуры (AVX для Intel, AMD; NEON для ARM) реализовать векторизованную версию матричного умножения (A*B = C)

Можно предполагать, что матрицы A, B - квадратные.

Обязательное требование - хранение всех матриц предполагается в едином порядке (все в row-major либо все в col-major).

Тип элементов матрицы - float для 128-битных, double для 256-битных векторных расширений.

Сравнить результаты и время выполнения векторизованного алгоритма с его последовательной версией. (N = 512, 1024, 2048).

Дедлайн: 21.10; 28.10

Для выполнения задания могут пригодиться:

(NEON): vdupq_n_f32, vmulq_f32, vaddq_f32

(AVX): _mm256_add_pd, _mm256_broadcast_pd, _mm256_mul_pd

При использовании SIMD и без него разный порядок действий. Чтобы минимизировать эффект (округления) от этого, используйте малые значения (хотя бы <1.0) при инициализации матриц.