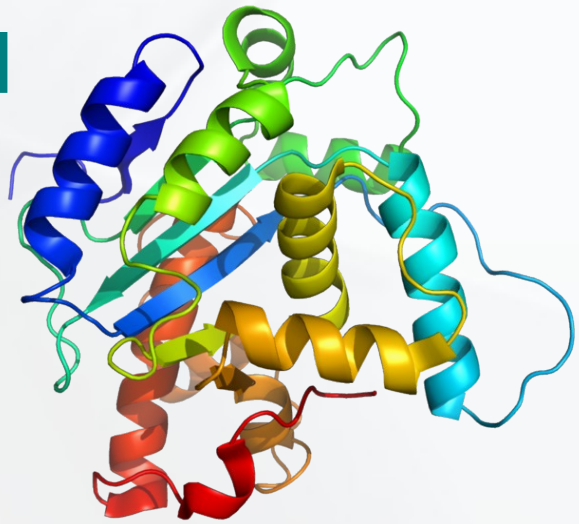
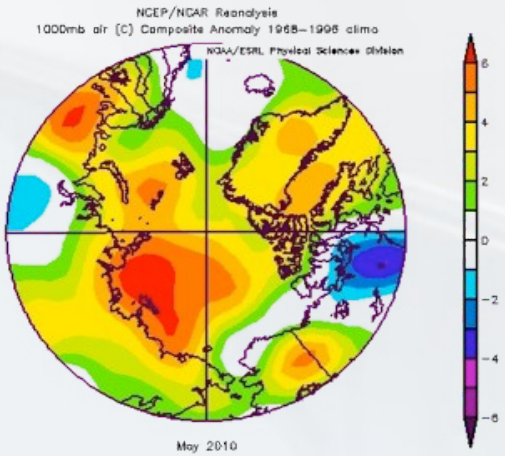
Параллельное программирование

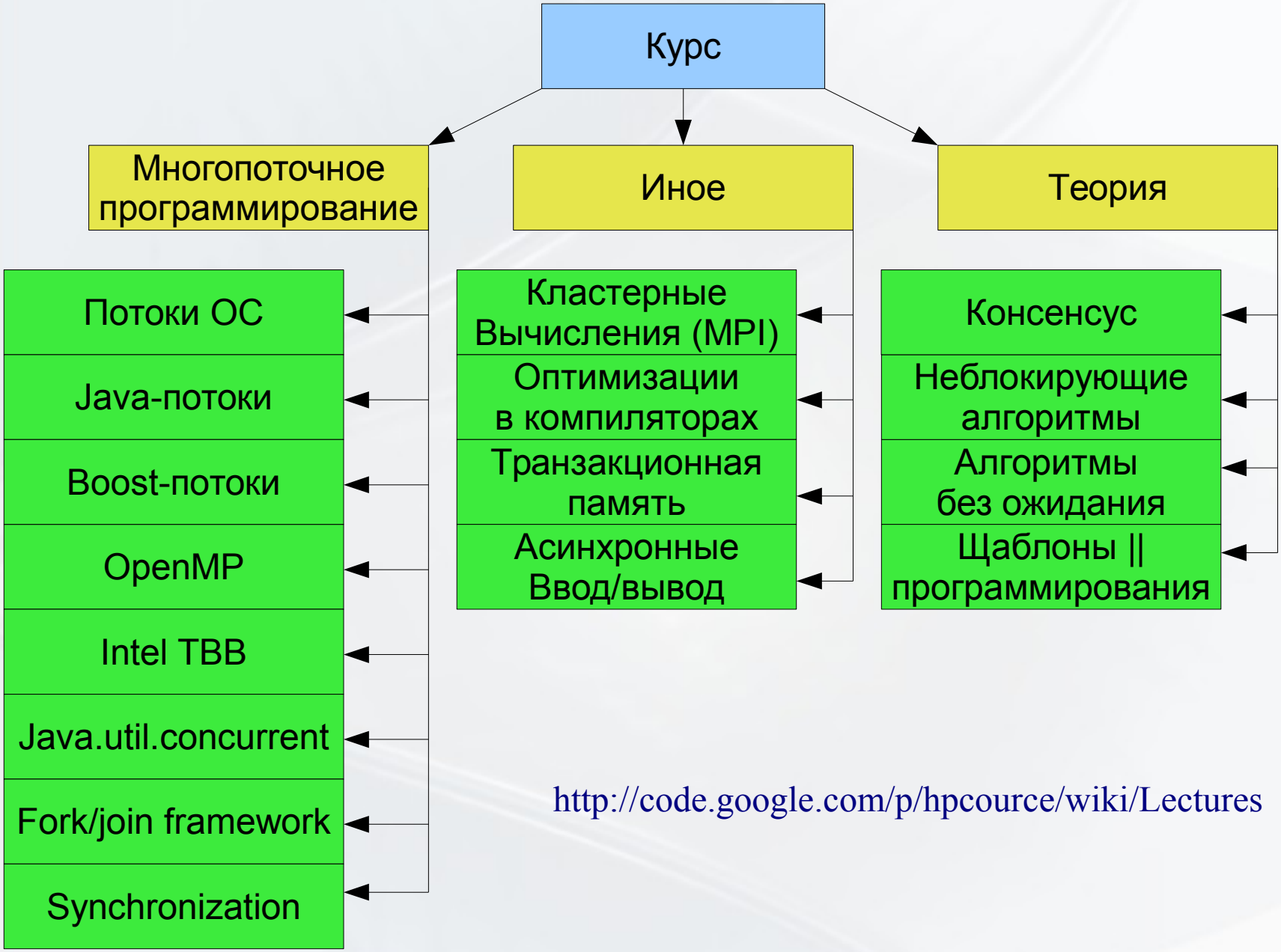
Калишенко Е.Л.
ПОМИ 2014

Мотивация



- Генетика и протеомика
- Климатология
- Физика высоких энергий
- Астрономия, банковские транзации...





http://code.google.com/p/hpcource/wiki/Lectures

SSE (Streaming SIMD Extensions)

Версия	Возможности
1	Восемь 128-битных регистров для 4 чисел по 32 бит (с плавающей точкой)
2	Теперь 2 64-битных числа в регистре
3	Уже 13 инструкций
4.1	47 инструкций (ускорение видео)
4.2	54 инструкции (операции со строками)

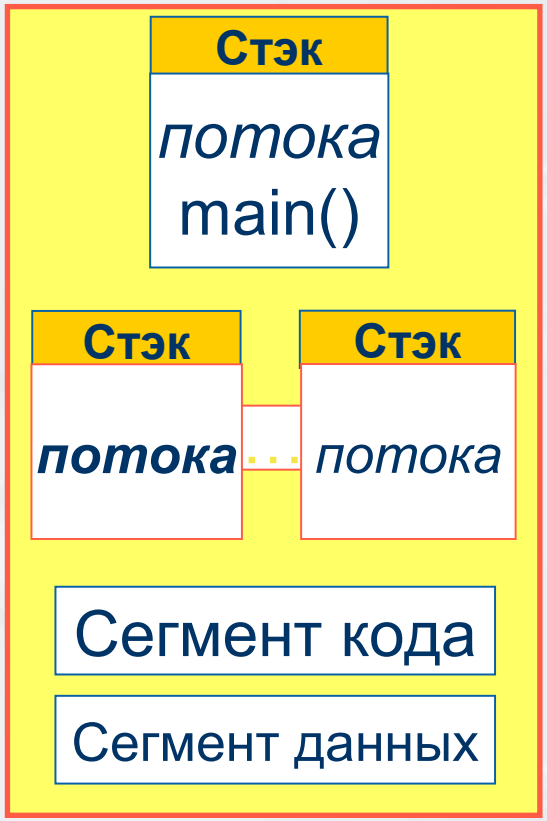
```
float a[4] = { 300.0, 4.0, 4.0, 12.0 };
float b[4] = {   1.5, 2.5, 3.5,  4.5 };

__asm {
    movups xmm0, a    ; // поместить из a в регистр xmm0
    movups xmm1, b    ; // поместить из b в регистр xmm1
    mulps xmm0, xmm1  ; // перемножить пакеты плавающих точек
    movups a, xmm0    ; // выгрузить результаты из регистра xmm0 по адресам a
};
```

Закон Амдала

- a — доля последовательного кода
- p — число процессоров

$$S = \frac{1}{a + \frac{(1-a)}{p}}$$

Процессы и потоки



- В начале выполнения процесс — 1 поток
- Потоки могут создавать новые в пределах одного процесса
- Все потоки имеют общие сегменты кода и данных
- Каждый поток имеет свой стек выполнения
- *Единица планирования ОС - поток*

Потоки ОС

Операция	Posix
Создание	pthread_create()
Ожидание завершения	pthread_join()
Захват мьютекса	pthread_mutex_lock()
Освобождение мьютекса	pthread_mutex_unlock()

```
int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine)(void*),
                   void *arg);
```

Java

- Наследование от Thread
- Реализация Runnable
- Остальное привычно: start(), join()

```
public class IntegrateRunnable implements Runnable {

        public IntegrateTask task;

        @Override
        public void run() {
                task.res = 0;
            for (double x = task.from; x < task.to - 1E-13*task.to; x += task.step) {
                task.res += task.f(x) * task.step;
            }
        }
}
```

Boost

Обёртка над потоками ОС (Posix или Win threads)

```
boost::thread_group th_group;


for(...i)
{
  boost::thread* th = new boost::thread(boost::bind(&Класс::<функция>,
                                                    this, i));
  th_group.add_thread(th);
}
th_group.join_all();
```

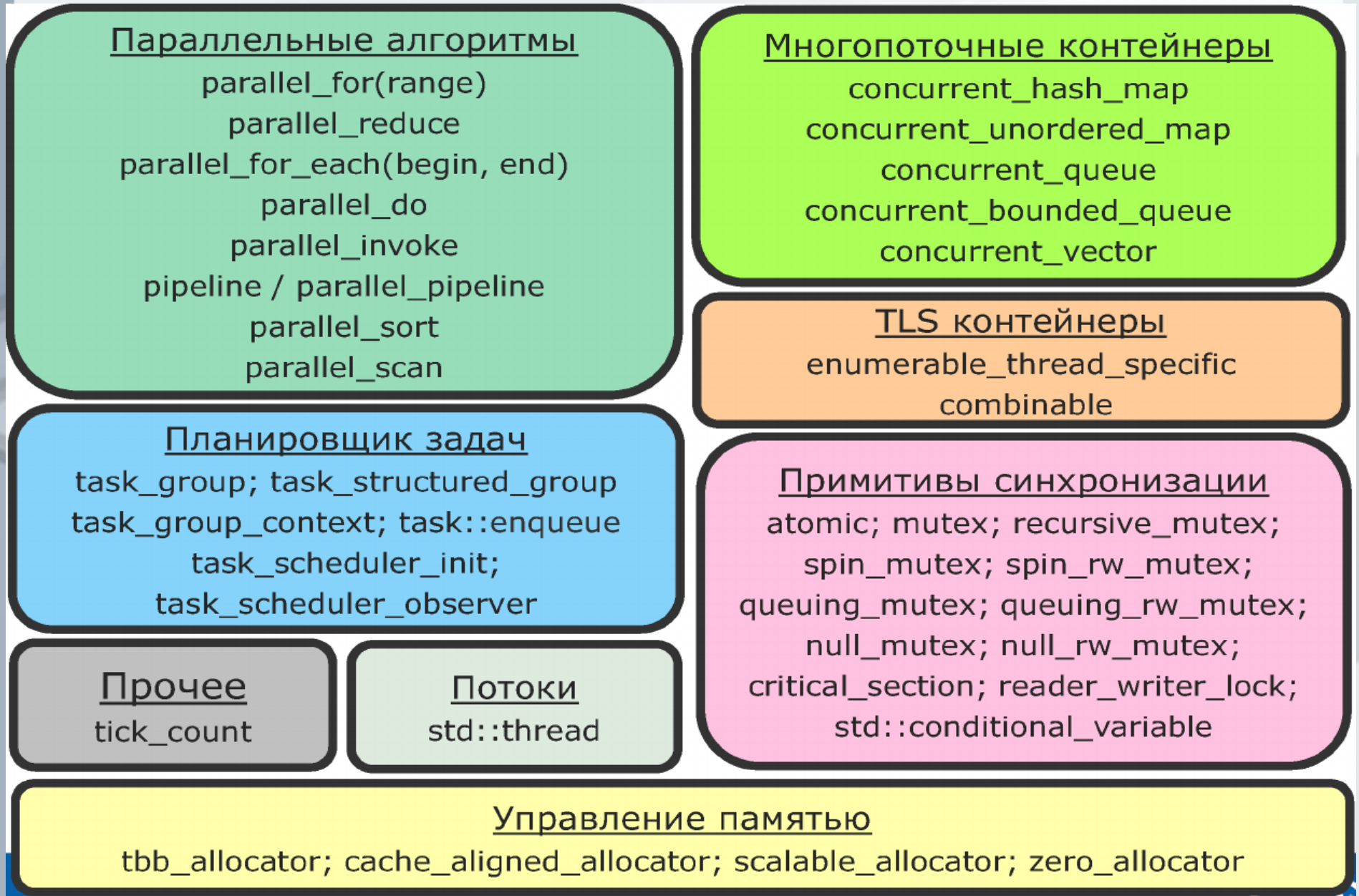
OpenMP

- Стандарт интерфейса для многопоточного программирования над общей памятью
- Набор средств для языков C/C++ и Fortran:
  - Директивы компилятора (#pragma omp ...)
  - Библиотечные подпрограммы (get_num_threads())
  - Переменные окружения (OMP_NUM_THREADS)

Пример OpenMP

```
#include <stdio.h>
#include <omp.h>
int main()
{
   int i;
   #pragma omp parallel
   {
      #pragma omp for
      for (i=0;i<1000;i++)
         printf("%d ",i);
   }
   return 0;
}
```

Intel TBB


Параллельные алгоритмы
parallel_for(range)
parallel_reduce
parallel_for_each(begin, end)
parallel_do
parallel_invoke
pipeline / parallel_pipeline
parallel_sort
parallel_scan
Многопоточные контейнеры
concurrent_hash_map
concurrent_unordered_map
concurrent_queue
concurrent_bounded_queue
concurrent_vector
TLS контейнеры
enumerable_thread_specific
combinable
Планировщик задач
task_group; task_structured_group
task_group_context; task::enqueue
task_scheduler_init;
task_scheduler_observer
Примитивы синхронизации
atomic; mutex; recursive_mutex;
spin_mutex; spin_rw_mutex;
queuing_mutex; queuing_rw_mutex;
null_mutex; null_rw_mutex;
critical_section; reader_writer_lock;
std::conditional_variable
Прочее
tick_count
Потоки
std::thread
Управление памятью
tbb_allocator; cache_aligned_allocator; scalable_allocator; zero_allocator

Примеры Intel TBB

```
void SortExample( ) {
   for( int i = 0; i < N; i++ ) {
      a[i] = sin((double)i);
      b[i] = cos((double)i);
   }
   parallel_sort(a, a + N);
   parallel_sort(b, b + N, std::greater<float>( ));
}
```

```
void ParallelApplyFoo(float a[], size_t n ) {
    parallel_for( blocked_range<size_t>( 0, n ),
     [&](const blocked_range<size_t>& range) {
       for(int i= range.begin(); i!=range.end(); i++)
             Foo(a[i]);
     },
     auto_partitioner() );
}
```

Java.util.concurrent

- Пулы потоков:
  - FixedThreadPool
  - CahcedThreadPool
  - SingleThreadPool...
- Атомики (AtomicBoolean, AtomicLong...)
- Потокобезопасные контейнеры
- Свои примитивы (ReentrantLock...)
- Будущее (Future<>)

Пример concurrent

```
private ExecutorService thread_pool;
private int[] array;

private class quick_sort_call implements Callable<Boolean>
{
...
  public Boolean call()
  {
    ...
    thread_pool.submit(...);
  }
}

public array_sorter() {
   thread_pool = Executors.newCachedThreadPool();
}

public void start_sorting()
{
   Future<Boolean> main_future=null;

   main_future = thread_pool.submit(new quick_sort_call(0, array.length-1, array));

   //waiting for the main future to be obtained...
   if(main_future!=null){
     try
     {
       if(main_future.get())System.out.println("Future obtained.");
     }catch(Exception ex)
     {
       System.out.println("Something wrong with multithread sorting! " + ex.getMessage());
     }
   }

   thread_pool.shutdown();
}
```