



**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  
**им. Н.И. Лобачевского**  
**- Национальный исследовательский университет -**





**Нижегородский государственный университет  
им. Н.И.Лобачевского  
– Национальный исследовательский университет –**

# **Современные тенденции в сфере высокопроизводительных вычислений**

**Сысоев А.В.**  
к.т.н, доцент каф. МОСТ  
института ИТММ

# **Часть I**

## **Вначале была цифра!**



# Считать всегда, считать везде...

---

- ☐ Что «умеют» делать современные компьютеры?
- ☐ Все (почти 😊)
- ☐ Игры, спецэффекты в фильмах, музыка, живопись, работа с текстами,...
  
- ☐ Что «умеют» делать современные компьютеры?
- ☐ Ничего, кроме вычислений
- ☐ Есть только целые и вещественные числа
  
- ☐ Как снимается противоречие?
- ☐ Задача – решение – алгоритм – программа



# Считать всегда, считать везде...



- ☐ «Синий» экран
- ☐ Захват изображения
- ☐ Триангуляция
- ☐ Трассировка лучей
- ☐ ...

- ☐ Для создания компьютерной модели T-1000 Роберта Патрика на протяжении 3-х дней снимали специальным трехмерным сканером
- ☐ После выхода фильма на экраны несколько членов «Федеральной лаборатории по испытанию атомного оружия» неофициально признали, что в данной картине содержится наиболее точное изображение ядерного взрыва, когда-либо показанного в художественных фильмах

# Считать всегда, считать везде



- ❑ Специальные модели с огромным числом параметров
- ❑ Сложные длительные расчеты
- ❑ Имманентная неопределенность
- ❑ Жесткое ограничение на время расчетов
- ❑ Предел предсказуемости (3-5-7 суток)

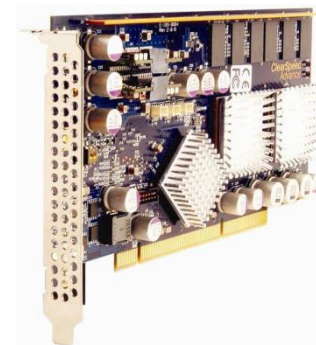
---

# **Часть II**

## **Суперкомпьютер суперкомпьютеру рознь**



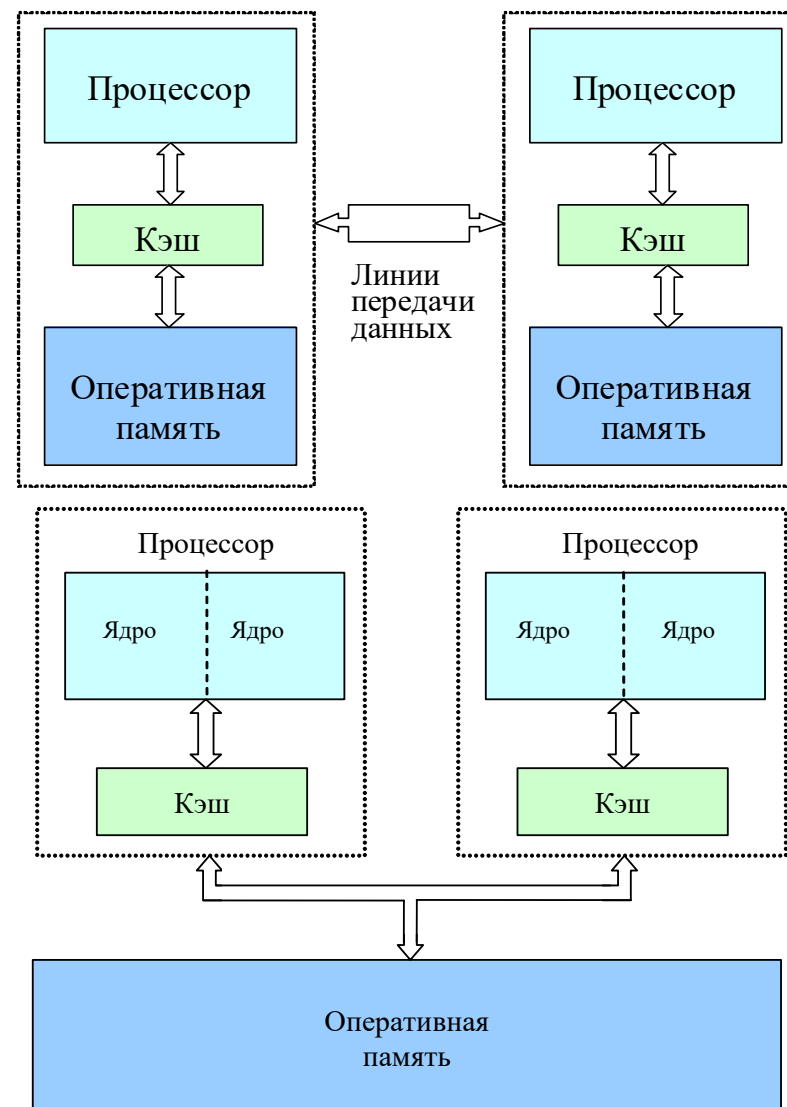
# От большого к малому...



# Классификация...

- ❑ Распределенная память
  - ❑ Кластерные системы

- ❑ Общая память
  - ❑ Многопроцессорные/многоядерные системы



- ❑ Гибридные вычислительные системы
  - ❑ Множество вычислительных узлов
  - ❑ Быстрая сеть между узлами
  - ❑ Несколько многоядерных процессоров в узле
  - ❑ Несколько ускорителей/сопроцессоров

# Суперкомпьютер – это...

---

## ❑ Что такое суперкомпьютер?

- ❑ Это вычислительная система, стоимость которой превышает 1 млн долларов
  - ❑ Это вычислительная система, производительность которой на 3 порядке выше, чем у современного настольного компьютера
  - ❑ Это вычислительная система, позволяющая решать задачи переднего края современной науки и техники
  - ❑ Это...
- 
- ❑ Это вычислительная система, на которой выполняются параллельные программы



# **Часть III**

## **Нам все параллельно**



# Делай раз, делай два...

---

## □ Алгоритм:

- точное предписание, задающее некоторый (например вычислительный) процесс
- начинается с произвольного исходного данного (из некоторой совокупности возможных данных)
- направлен на получение результата
- результат полностью определяется исходным данным

□ «Пойди туда, не знаю куда, принеси то, не знаю что» – алгоритм?



# Делай раз, делай два

---

- ❑ Что такое **последовательный** алгоритм?
  - ❑ алгоритм, выполнение которого предполагается строго последовательным
  
- ❑ Чисто последовательный алгоритм
  - ❑ алгоритм, все действия которого могут выполняться **только в том порядке**, в котором они указаны



- ❑ Что такое **параллельный** алгоритм?
  - ❑ алгоритм, часть действий которого может выполняться одновременно (**параллельно**) при наличии нескольких исполнителей
  
- ❑ Существуют ли чисто параллельные алгоритмы?
  
- ❑ **Распараллеливание**
  - ❑ процесс формулировки алгоритма с явным выделением участков, допускающих параллельное выполнение

# Программа-минимум

---

## ❑ Что такое **программа**?

- ❑ Программа – упорядоченная последовательность действий для ЭВМ, реализующая алгоритм решения некоторой задачи

## ❑ Что такое **последовательная программа**?

- ❑ Последовательная программа – реализация **последовательного** алгоритма

## ❑ Что такое **последовательная программа**?

- ❑ Последовательная программа – реализация алгоритма, рассчитанная на наличие **только** одного исполнителя



# Программа-максимум

---

- ❑ Что такое **параллельная программа**?
  - ❑ Параллельная программа – реализация алгоритма, **рассчитанная на наличие более чем одного исполнителя**
  - ❑ Параллельная программа – это программа, **некоторые участки которой выполняются одновременно**



# Зачем они нужны

## □ Показатель эффективности

- Пусть  $T_1$  – время выполнения последовательного алгоритма при использовании одного процессора
- Пусть  $T_p$  – время выполнения параллельного алгоритма при использовании  $p$  процессоров
- Тогда Ускорение  $S_p$  параллельного алгоритма к последовательному есть

$$S_p = T_1/T_p$$

(!) В наилучшем случае  $S_p = p$

(?) Достижимо ли наилучшее ускорение ( $p$ )



# Закон Амдаля

- ❑ (!) Наилучшее ускорение ( $p$ ) недостижимо
- ❑ Ускорение  $S_p$  параллельного алгоритма с долей последовательных вычислений равной  $f$  при наличии  $p$  процессоров не может быть больше

$$S_p \leq \frac{1}{f + (1 - f) / p},$$

- ❑ (!) Если  $f = 0.1$ , то  $S_p$  не может быть больше 10-ти.



# **Часть IV**

## **Сага об инструментах**



# Ничто так не ограничивает полет мысли программиста, как компилятор...

---

- ❑ Для разработки программ требуются:
  - ❑ Языки программирования
  - ❑ Редакторы кода (текстовые редакторы)
  - ❑ Компиляторы, компоновщики
- ❑ Для разработки программ полезны:
  - ❑ Отладчики
  - ❑ Инструменты тестирования
  - ❑ Системы контроля версий
  - ❑ Библиотеки функций
  - ❑ ...



# Огласите весь список, пожалуйста...

---

- ❑ Для кластерных систем:
  - ❑ Языки: C\C++, Fortran
  - ❑ Технология MPI (Message Passing Interface)
  - ❑ Системы сбора, построения и анализа трасс
  - ❑ ...
- ❑ Для систем с общей памятью
  - ❑ Языки: C\C++, Fortran
  - ❑ Технологии OpenMP, Cilk Plus, TBB
  - ❑ Intel Parallel Studio (Inspector, Amplifier)
  - ❑ ...



# **Часть V**

## **Лучше один раз увидеть...**



# Скалярное произведение векторов

- $a = (a_1, a_2, \dots, a_n), b = (b_1, b_2, \dots, b_n)$
- $c = (a, b) = a_1 * b_1 + a_2 * b_2 + \dots + a_n * b_n$
- Последовательная реализация

```
const int size = 1000000;  
double a[size], b[size];  
double c;  
int i;  
  
c = 0;  
for (i = 0; i < size; i++)  
    c += a[i] * b[i];
```



# Скалярное произведение векторов

- Параллельная реализация (технология OpenMP)

```
const int size = 1000000;  
double a[size], b[size];  
double c;  
int i;  
  
c = 0;  
#pragma omp parallel for reduction(+:c)  
for (i = 0; i < size; i++)  
    c += a[i] * b[i];
```



# Скалярное произведение векторов

## □ Параллельная реализация (технология MPI)

```
const int size = 1000000;  
double a[size], b[size];  
double c, _c;  
int i, myid, numprocs;  
MPI_Status status;  
  
MPI_Init(&argc, &argv);  
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);  
MPI_Comm_rank(MPI_COMM_WORLD, &myid);  
MPI_Scatter(a, n / numprocs, MPI_DOUBLE, a, n / numprocs,  
           MPI_DOUBLE, 0, MPI_COMM_WORLD);  
MPI_Scatter(b, n / numprocs, MPI_DOUBLE, b, n / numprocs,  
           MPI_DOUBLE, 0, MPI_COMM_WORLD);  
_c = 0;  
for (i = 0; i < n; i++)  
    _c += a[i] * b[i];  
MPI_Reduce(&_amp;c, &c, 1, MPI_DOUBLE, MPI_SUM, 0,  
MPI_COMM_WORLD);
```

# **Часть Финальная**

## **Show must go on!**



---

# Трек

## **«Параллельное программирование для систем с общей и распределенной памятью»**



*Вопросы?*

