

Использование графических ускорителей при решении задач обработки текстов

Афонин С.А.
Сыроватский Д.А.

1.11.2011
МГУ им.М.В.Ломоносова

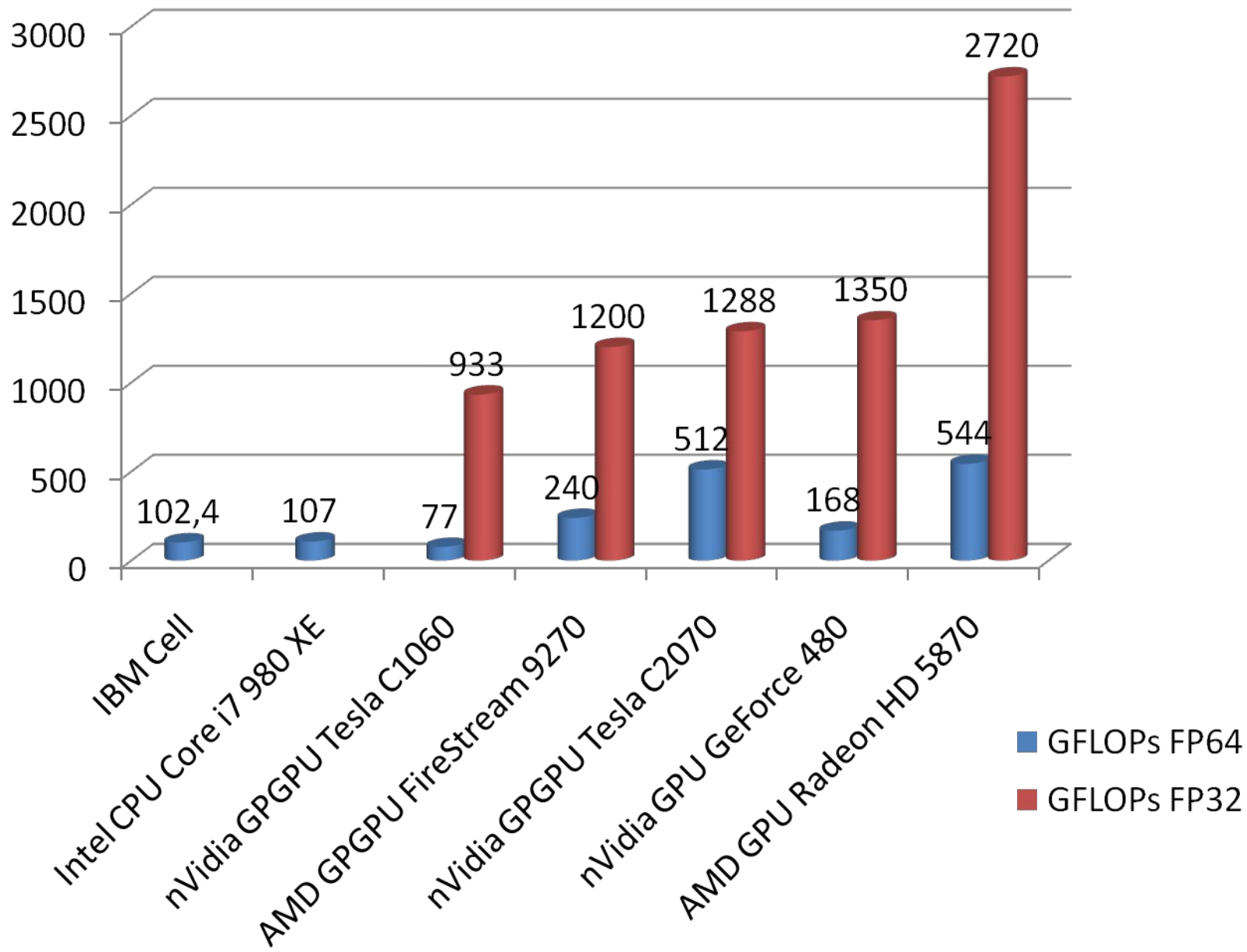
План

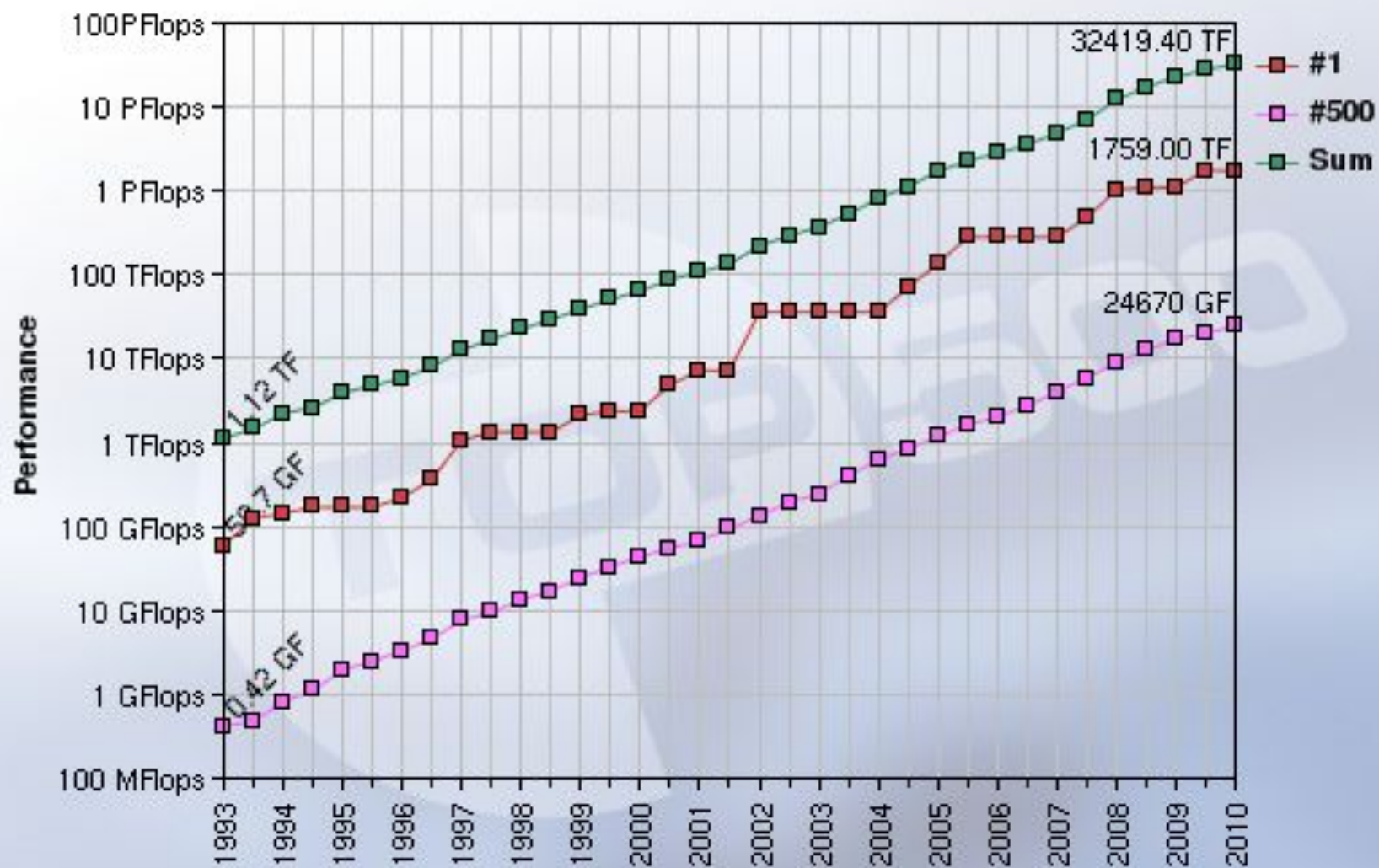
- Что такое GPU и CUDA
- Алгоритмы анализа данных
- Задачи обработки текстов

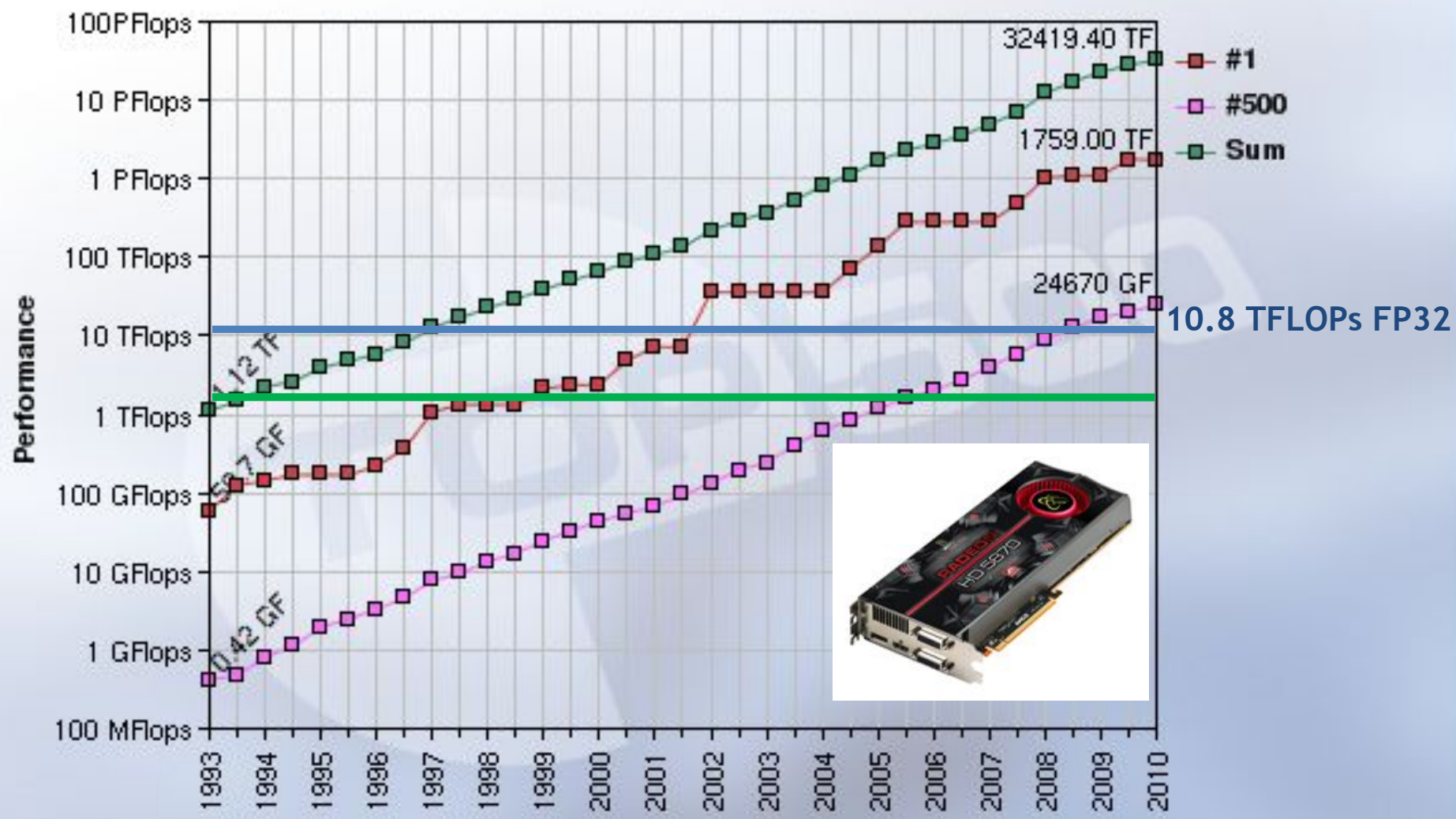
GPU и CUDA

- GPU = Graphic Processing Unit
- CUDA = Computing Unified Device Architecture

Почему графические
ускорители (GPU)?







Внешний вид



Графические процессоры





#2 in Top500: NEBULAE

1.27 PFlops Linpack 2.9 PFlops peak

曙光星云高效能计算机系统
DAWNING NEBULAE



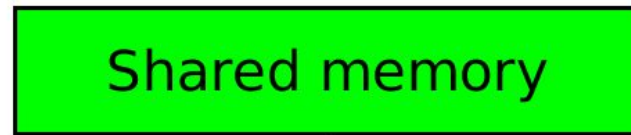
CUDA – почти C

- Единственное отличие – добавления для работы с потоками

Архитектура CUDA

- SIMD мультипроцессоры (8 или 16 ядер)
- Мультипроцессор имеет регистры и разделяемую (локальную) память
- Задача разбивается на блоки, блоки — на потоки
- Блоки назначаются на процессоры; выполненный блок невозможно запустить повторно

Общая для элементов блока



Персональная для
элемента блока



```
[talis@tesla ~]$ NVIDIA_GPU_Computing_SDK/C/bin/linux/release/deviceQuery
CUDA Device Query (Runtime API) version (CUDA static linking)
There is 1 device supporting CUDA
```

```
Device 0: "Tesla C1060"
```

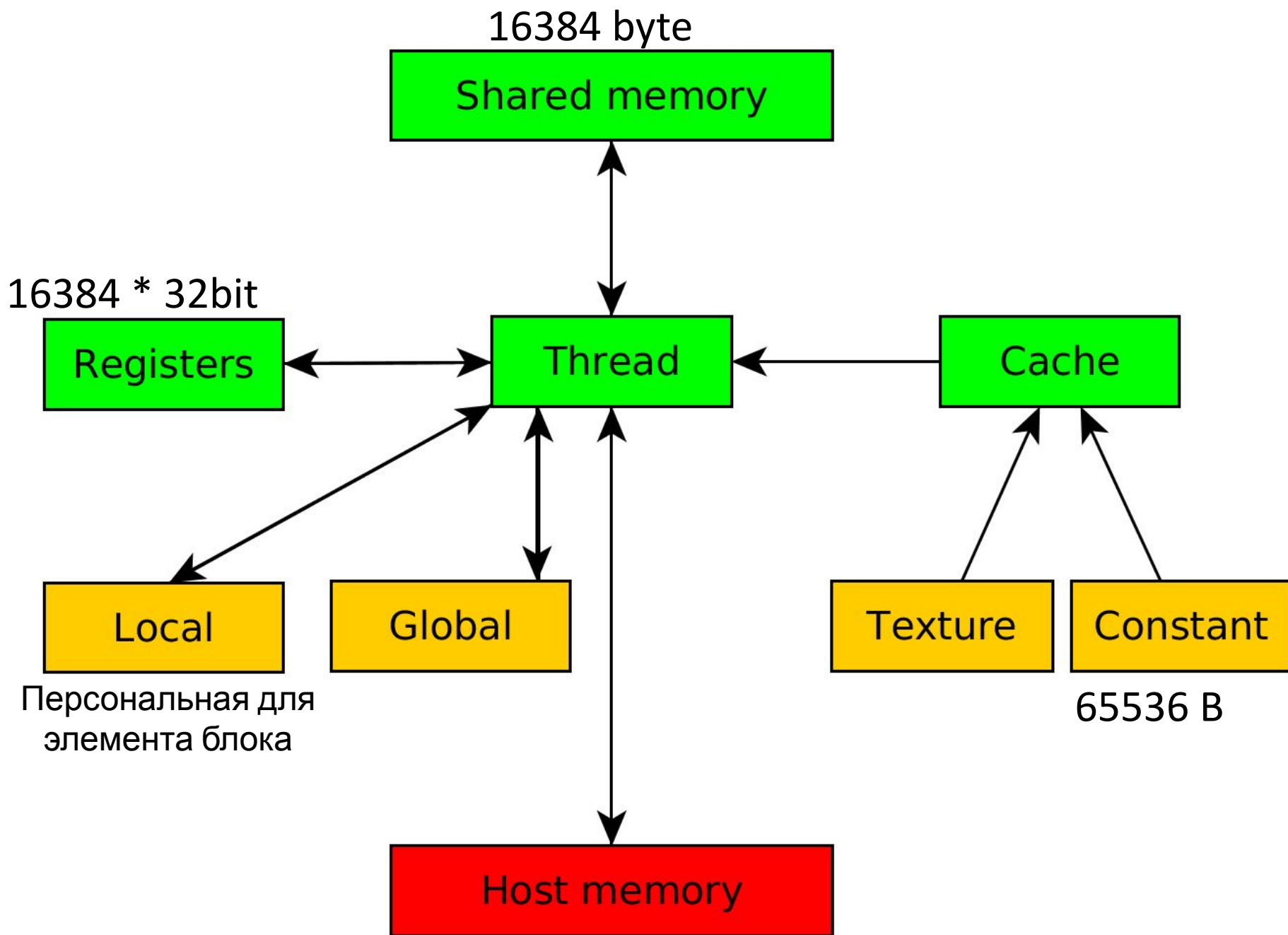
```
  CUDA Driver Version:            3.10
  CUDA Runtime Version:          3.10
  CUDA Capability Major revision number: 1
  CUDA Capability Minor revision number: 3
  Total amount of global memory:  4294770688 bytes
  Number of multiprocessors:      30
  Number of cores:                240
  Total amount of constant memory: 65536 bytes
  Total amount of shared memory per block: 16384 bytes
  Total number of registers available per block: 16384
  Warp size:                      32
  Maximum number of threads per block: 512
  Maximum sizes of each dimension of a block: 512 x 512 x 64
  Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
  Maximum memory pitch:           2147483647 bytes
  Texture alignment:              256 bytes
  Clock rate:                     1.30 GHz
  Concurrent copy and execution:   Yes
  Run time limit on kernels:       No
  Integrated:                      No
  Support host page-locked memory mapping: Yes
  Compute mode:                   Default (multiple host threads
```

```
can use this device simultaneously)
```

```
Test PASSED
```

```
Press ENTER to exit...
```

```
[talis@tesla ~]$ █
```



Расширения C для CUDA

Стандартный C код

```
void saxpy_serial(int i,
float a, float *x, float *y)
{
    y[i] = a*x[i] + y[i];
}

// Вызов последовательного
// ядра SAXPY

for (int i = 0; i < n; ++i)
    saxpy_serial(i, 2.0, x, y);
```

Код на CUDA C

```
__global__
void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n) y[i] = a*x[i] + y[i];
}

// Вызов параллельного ядра SAXPY с размером
// блока в 256 потоков
int nblocks = (n + 255) / 256;

saxpy_parallel<<<nblocks, 256>>>(n, 2.0f, x, y);
```

Алгоритмы анализа данных

- Выявление ассоциативных зависимостей (Association rule mining, Apriori)
- Классификация (KNN)
- Кластеризация (K-means)
- Уменьшение размерности данных

•Выявление зависимостей

- $I = \{i_1, \dots, i_m\}$ — множество атрибутов
- База данных — набор записей вида (TID, i_1, \dots, i_p)
- Частотный k -набор — k -подмножество I , элементы которого встречаются более чем в N записях
- Задача: найти все частотные k -наборы
- Зависимости: если набор содержит X , то он содержит и x' с вероятностью p

Алгоритм выявления

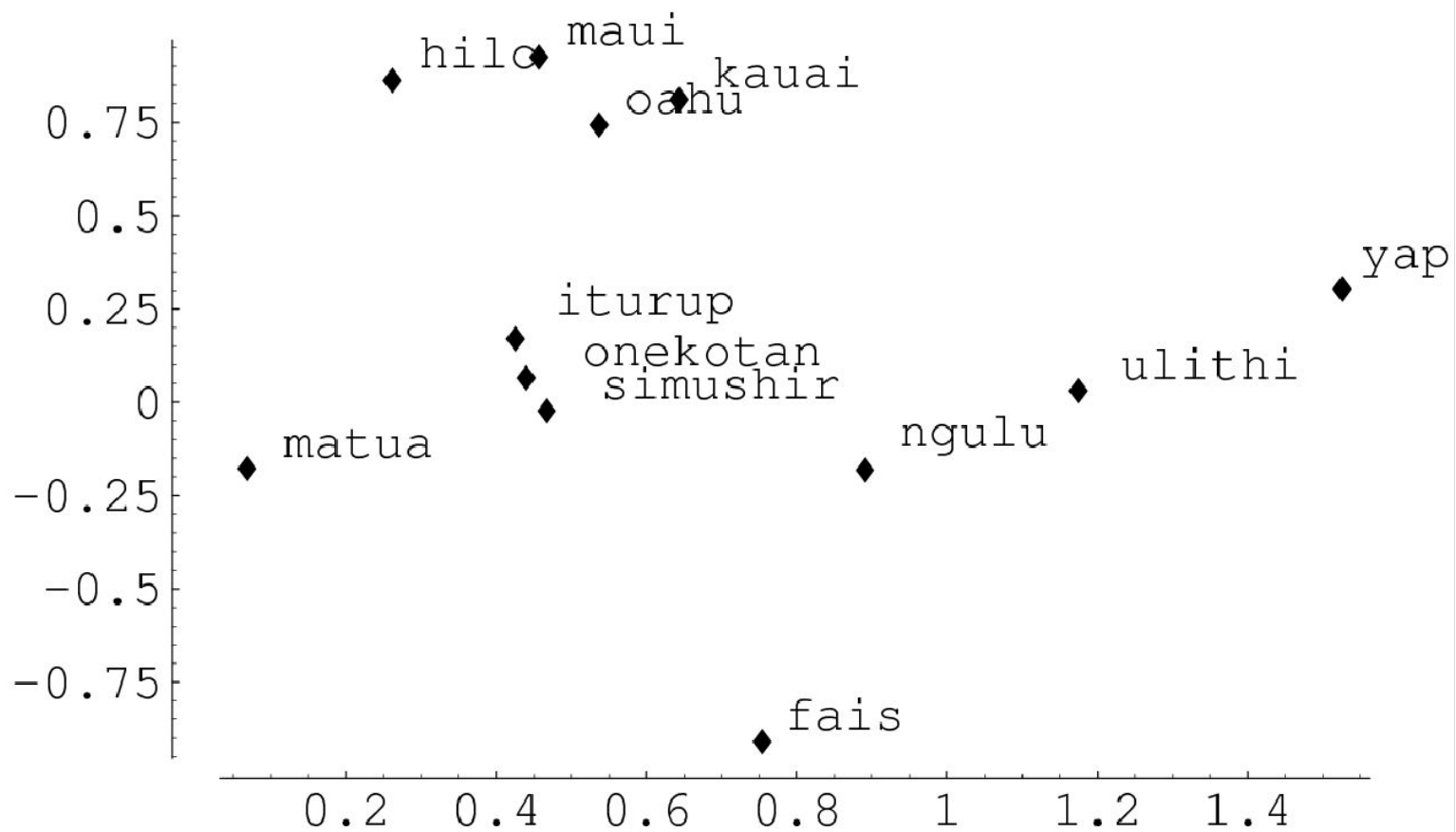
- Найти все частотные 1-наборы
- Для $k=2, \dots$ и пока есть новые наборы
 - Построение k -кандидатов: объединение двух частотных $(k-1)$ -наборов с общим $(k-2)$ -префиксом
 - Фильтрация: k -кандидат удаляется, если он содержит не частотное $(k-1)$ подмножество
 - Определение частотности кандидатов

Классификация

- Метод ближайших соседей
 - Задана выборка объектов с приписанными метками
 - Для нового объекта вычисляется расстояние до всех объектов выборки
 - Метка нового объекта — самая частотная метка его K ближайших соседей из выборки

Понижение размерности

- На вход алгоритма поступает матрица расстояний, принцип действия следующий:
- На плоскости случайным образом фиксируются точки, попарно соединенные пружинами, длины ненапряженных состояний которых берутся из матрицы расстояний. Затем точки отпускаются, и действующие на них силы приводят потенциальную энергию системы к минимуму. Находятся варианты расположения точек, приводящие к минимуму потенциальной энергии и (или) лучше других удовлетворяющие другим формулам оценки качества распределения.
- Например, если матрица расстояний строилась по точкам, лежащим на плоскости, то в двумерное пространство точки восстановятся с точностью до поворота и смены знаков осей



Производительность на GPU: тысячи точек за секунды

