Лаборатория синтеза параллельных программ ИВМиМГ СО РАН
Кафедра параллельных вычислительных технологий НГТУ
Кафедра параллельных вычислений НГУ

ЭФФЕКТИВНОЕ ПРОГРАММИРОВАНИЕ СОВРЕМЕННЫХ МИКРОПРОЦЕССОРОВ

Киреев Сергей Евгеньевич

ИВМиМГ СО РАН

XXXI Летняя школа по параллельному программированию, 4 июля 2018 г.

ОПТИМИЗАЦИЯ ПРОГРАММ

Цель оптимизации:

- Программа должна работать как можно быстрее

- При этом хотим затратить как можно меньше усилий
  - на изучение способов оптимизации
  - на саму оптимизацию

ОПТИМИЗАЦИЯ ПРОГРАММ

- Выделим ТРИ УРОВНЯ вникания в проблему оптимизации программ
  1. Первый: минимальный
  2. Второй: приемлемый в большинстве случаев
  3. Третий: продвинутый

ПЛАН ДОКЛАДА

- Первый уровень вникания в проблему
- Второй уровень вникания в проблему
- Третий уровень вникания в проблему

ПЕРВЫЙ УРОВЕНЬ

- Принцип: воспользоваться готовыми средствами

- Готовые средства:
  - Оптимизирующий компилятор
  - Библиотеки оптимизированных подпрограмм

ПЕРВЫЙ УРОВЕНЬ. КОМПИЛЯТОР

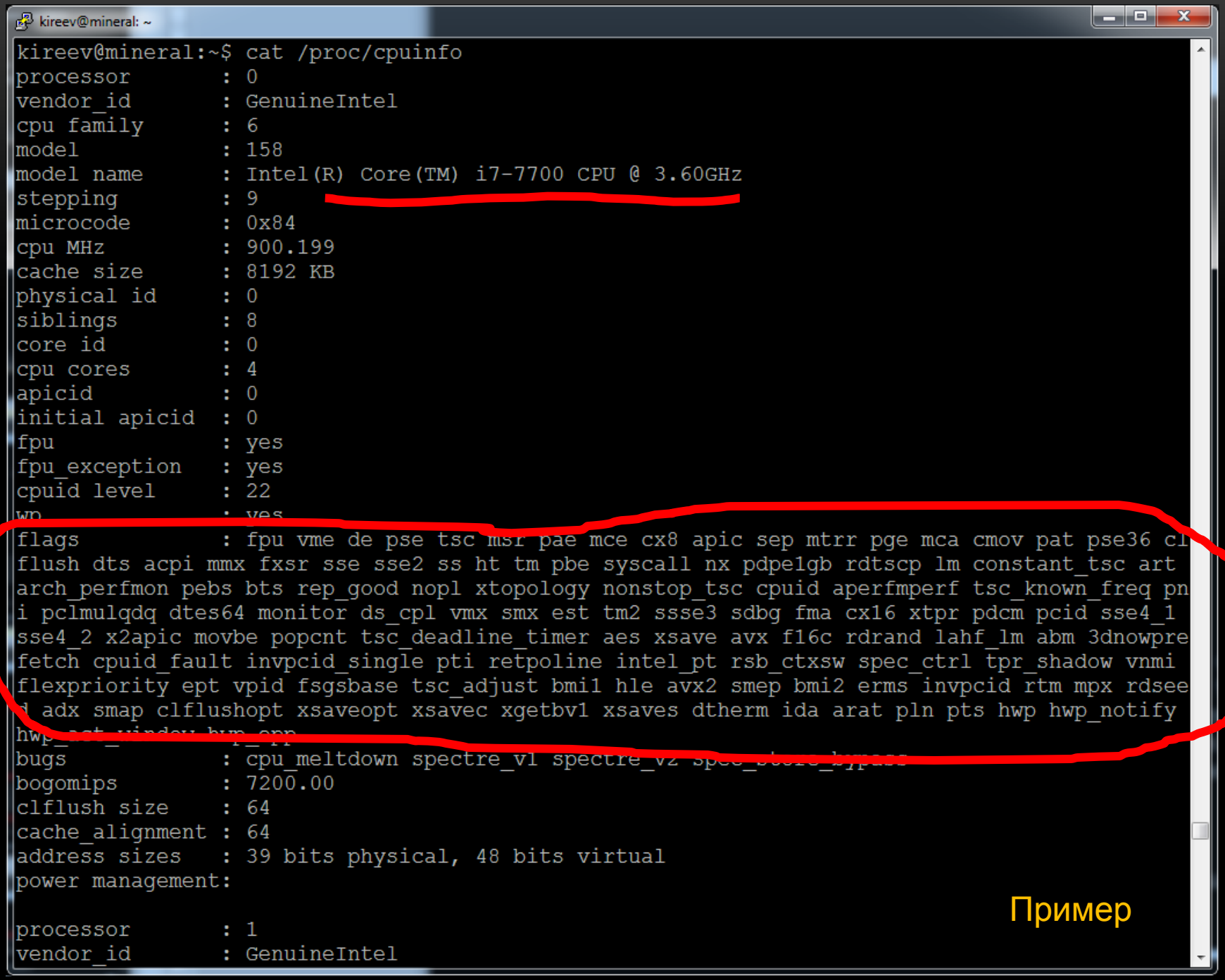
- Многие компиляторы уже умеют хорошо оптимизировать программы
- Что нужно знать, чтобы использовать компилятор для оптимизации?
  - Целевой процессор: название, архитектура, расширения
  - Доступные компиляторы и как ими пользоваться (ключи, опции)
- Как получить эту информацию?
  - cat /proc/cpuinfo, CPU-Z, …
  - man gcc, man icc, clang --help, …
  - Google

kireev@mineral: ~

```
kireev@mineral:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
stepping        : 9
microcode       : 0x84
cpu MHz         : 900.199
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
flush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pn
i pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpre
fetch cpuid_fault invpcid_single pti retpoline intel_pt rsb_ctxsw spec_ctrl tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdsee
d adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify
hwp_act_window hwp_epp
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips        : 7200.00
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
```

Пример


kireev@mineral: ~
kireev@mineral:~$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 158
model name : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
stepping : 9
microcode : 0x84
cpu MHz : 900.199
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 0
cpu cores : 4
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 22
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
flush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pn
i pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpre
fetch cpuid_fault invpcid_single pti retpoline intel_pt rsb_ctxsw spec_ctrl tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdsee
d adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify
hwp_act_window hwp_epp
bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips : 7200.00
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
processor : 1
vendor_id : GenuineIntel
Пример

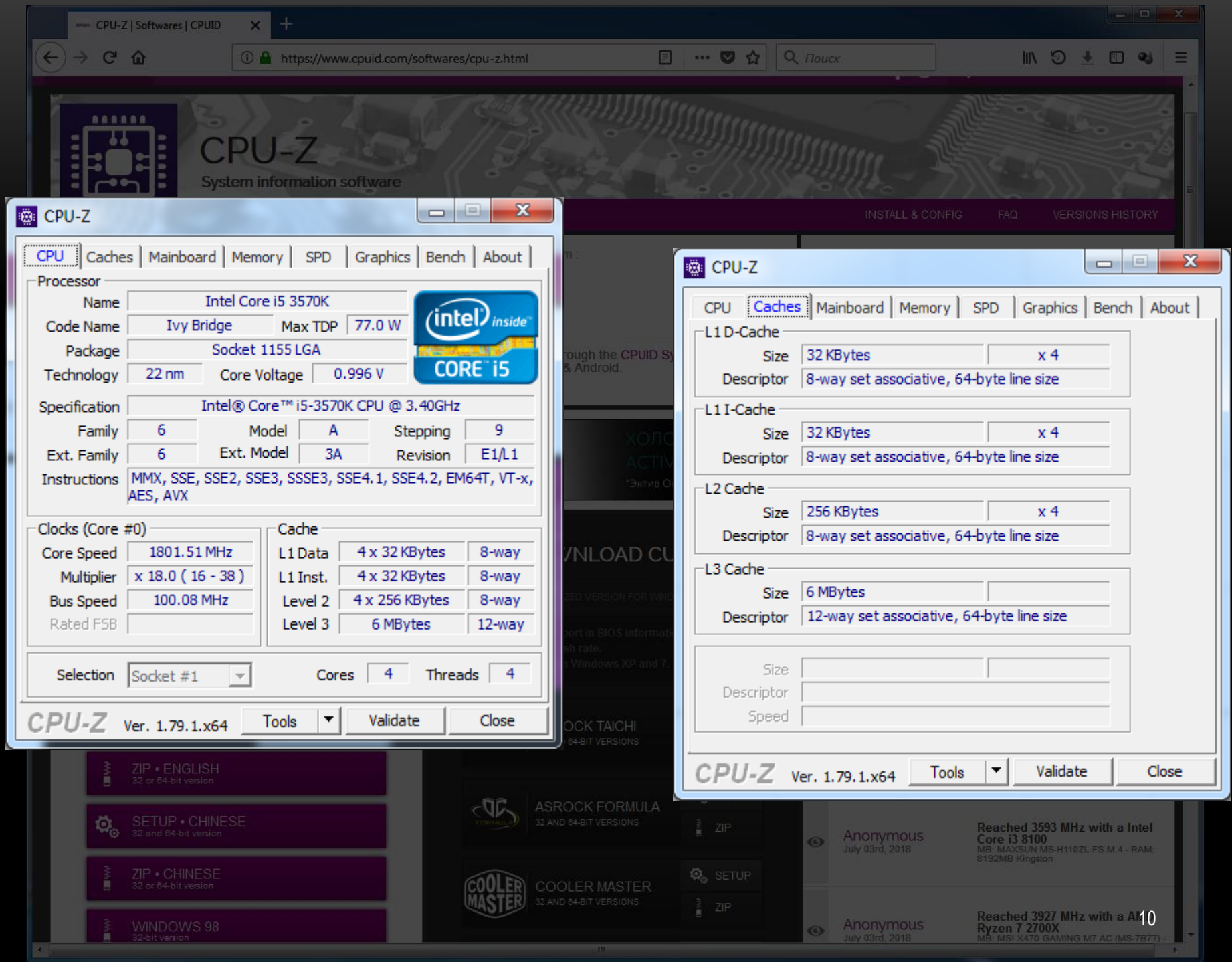
kireev@mineral: ~

```
kireev@mineral:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
stepping        :
microcode       :
cpu MHz         : 900.199
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
flush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pn
i pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpre
fetch cpuid_fault invpcid_single pti retpoline intel_pt rsb_ctxsw spec_ctrl tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdsee
d adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify
hwp_act_window hwp_epp
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips        : 7200.00
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
```

https://en.wikipedia.org/wiki/List_of_Intel_Core_i7_microprocessors

Пример


CPU-Z | Softwares | CPUID
https://www.cpuid.com/softwares/cpu-z.html
CPU-Z
System information software
INSTALL & CONFIG
FAQ
VERSIONS HISTORY
CPU-Z
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
Processor
Name
Intel Core i5 3570K
Code Name
Ivy Bridge
Max TDP
77.0 W
Package
Socket 1155 LGA
Technology
22 nm
Core Voltage
0.996 V
Specification
Intel® Core™ i5-3570K CPU @ 3.40GHz
Family
6
Model
A
Stepping
9
Ext. Family
6
Ext. Model
3A
Revision
E1/L1
Instructions
MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX
Clocks (Core #0)
Core Speed
1801.51 MHz
Multiplier
x 18.0 (16 - 38)
Bus Speed
100.08 MHz
Rated FSB
Cache
L1 Data
4 x 32 KBytes
8-way
L1 Inst.
4 x 32 KBytes
8-way
Level 2
4 x 256 KBytes
8-way
Level 3
6 MBytes
12-way
Selection
Socket #1
Cores
4
Threads
4
CPU-Z
Ver. 1.79.1.x64
Tools
Validate
Close
CPU-Z
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
L1 D-Cache
Size
32 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L1 I-Cache
Size
32 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L2 Cache
Size
256 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L3 Cache
Size
6 MBytes
Descriptor
12-way set associative, 64-byte line size
Size
Descriptor
Speed
CPU-Z
Ver. 1.79.1.x64
Tools
Validate
Close
ZIP • ENGLISH
SETUP • CHINESE
ZIP • CHINESE
WINDOWS 98
ASROCK FORMULA
COOLER MASTER
Anonymous
July 03rd, 2018
Reached 3593 MHz with a Intel Core i3 8100
Anonymous
July 03rd, 2018
Reached 3927 MHz with a AM Ryzen 7 2700X

ПЕРВЫЙ УРОВЕНЬ. КОМПИЛЯТОР

- Какие бывают компиляторы?
  - Intel C/C++ Compiler
  - GNU Compiler Collection
  - LLVM/Clang
  - AMD x86 Open64 Compiler Suite
  - Microsoft C/C++ Compiler
  - …
- Как правильно воспользоваться компилятором?
  - Задать необходимые ключи оптимизации
  - Использовать оптимизацию на основе сбора статистики

«A Performance-Based Comparison of C/C++ Compilers» by Colfax Research

ПЕРВЫЙ УРОВЕНЬ: КОМПИЛЯТОР

- Ключи, задающие уровень оптимизации:
  - -O0, -O1, -O2, -O3, -O, -Osize, -Ofast, …
- Ключи, задающие целевую архитектуру, расширения системы команд
  - -march=corei7, -mavx2, -fma, …
- Ключи, задающие использование более быстрой и менее точной арифметики
  - -ffast-math, -fast-transcendentals, -fp-model fast=2
- Ключи, указывающие, что программа соответствует «ANSI aliasing rules»
  - -ansi-alias, -fstrict-aliasing

ПЕРВЫЙ УРОВЕНЬ: КОМПИЛЯТОР

- Оптимизация на основе сбора статистики

1. Компиляция программы с добавлением кода для сбора статистики
   - **$g++ -O1 -fprofile-generate -o prog prog.c**
   - Создаётся файл: prog
2. Запуск программы много раз на типичных данных – сбор статистики
   - **$./prog**
   - Создаётся файл: prog.gcda
3. Компиляция программы с оптимизацией с учётом статистики
   - **$ g++ -O1 -fprofile-use -o prog_opt prog.c**
   - Создается файл: prog_opt

Пример (хороший)

```
$time ./cache
real    0m40.409s
user    0m40.323s
sys     0m0.016s
$time ./cache_opt
real    0m28.768s
user    0m28.710s
sys     0m0.008s
```

ОР

статистики

р статистики

стики

- **$ g++ -O1 -fprofile-use -o prog_opt prog.c**
- Создается файл: prog_opt

Пример (хороший)

```
$time ./cache
real    0m40.409s
user
sys
$time .
real
user
sys
```

Пример (плохой)

```
$time ./wave
real    0m12.442s
user    0m12.439s
sys     0m0.004s
$time ./wave_opt
real    0m15.714s
user    0m15.706s
sys     0m0.008s
```

- $ g++
- Созд

ПЕРВЫЙ УРОВЕНЬ: БИБЛИОТЕКИ

Многие стандартные операции уже эффективно реализованы в различных библиотеках

Что нужно знать:

1. Какие стандартные операции есть в программе?
2. Какие есть библиотеки, реализующие эти операции?
3. Как правильно настроить библиотеку (задать параметры)?
   - параметры процессора, ускорителей
   - наличие компиляторов, других библиотек, …

ПЕРВЫЙ УРОВЕНЬ: БИБЛИОТЕКИ

- Примеры библиотек по предметным областям:
  - BLAS, LAPACK, Eigen – операции линейной алгебры
  - FFTW – быстрое преобразование Фурье
  - PETSc – решение дифференциальных уравнений в частных производных, решение СЛАУ итерационными методами
  - PARDISO, SuperLU – решение СЛАУ прямым методом
  - GMP – арифметика произвольной точности
  - Intel DAAL – Data Analytics Acceleration Library – анализ данных
  - TensorFlow – машинное обучение
  - …
- Примеры библиотек по производителям:
  - Intel MKL – Math Kernel Library, Intel IPP – Integrated Performance Primitives, Intel DAAL – Data Analytics Acceleration Library
  - ACML – AMD Core Math Library
  - NVIDIA GPU-Accelerated libraries
  - Rogue Wave IMSL – International Mathematics and Statistics Library
  - NAG library
  - …

ПЛАН ДОКЛАДА

- Первый уровень вникания в проблему:
  - Компилятор + библиотеки
- Второй уровень вникания в проблему
- Третий уровень вникания в проблему

ВТОРОЙ УРОВЕНЬ

- Принцип: учесть самые важные вопросы, касающиеся скорости работы программы

- Самые важные вопросы:
  - Скорость доступа к данным в памяти
  - Векторизация вычислений
  - Использование всех ядер – распараллеливание

ВТОРОЙ УРОВЕНЬ

- Что нужно знать на втором уровне?

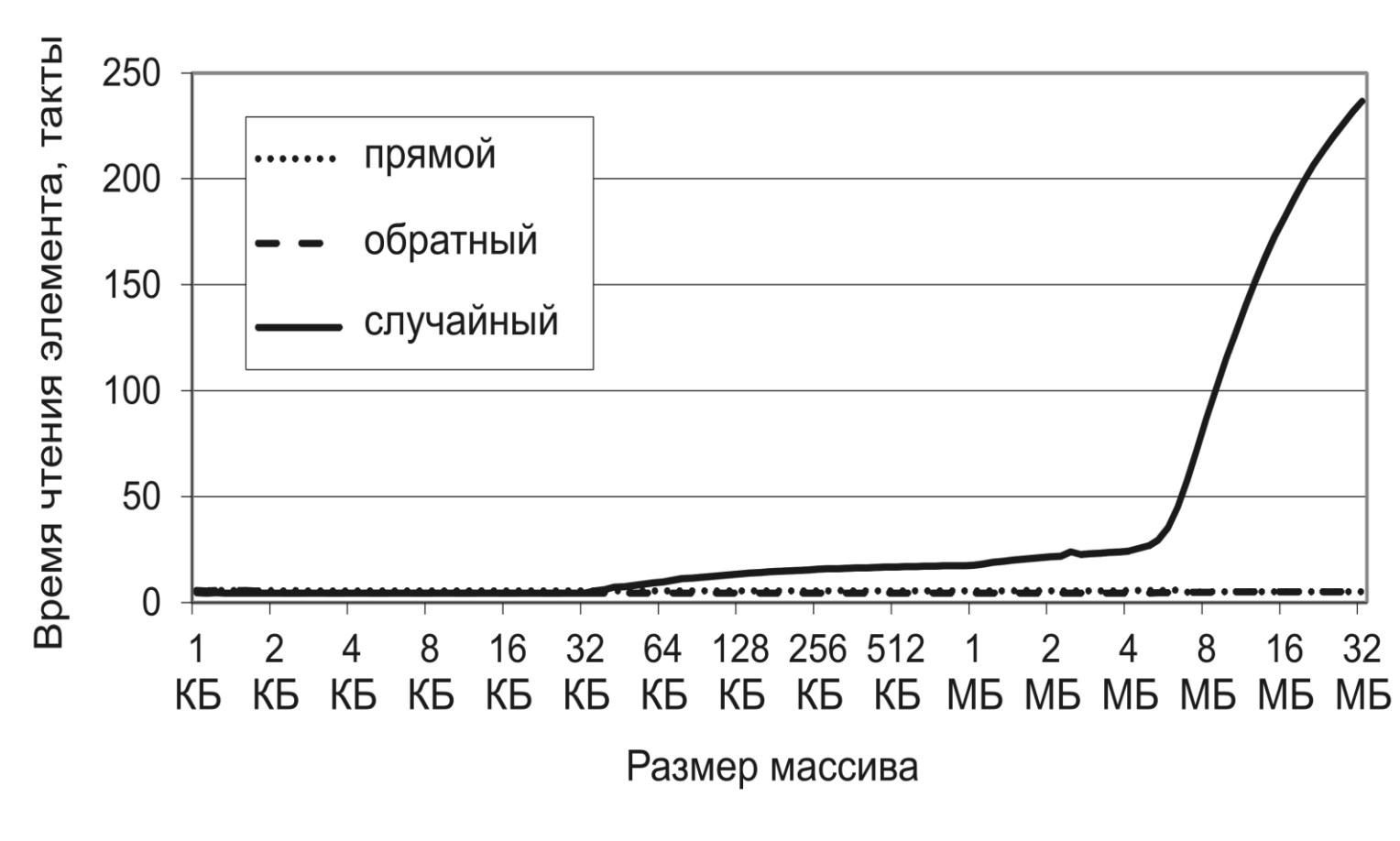
1. Доступ к памяти намного дольше, чем большинство арифметических операций
   - add : 3 такта, mul : 5 тактов, div : 20 тактов, sqrt : 21 такт
   - чтение из кэша L1: 4 такта, чтение из памяти : 50-200 тактов
2. Процессоры имеют векторные операции, которые могут делать несколько скалярных операций одновременно
   - 2, 4, 8 операций одновременно
3. Процессоры многоядерные. Чтобы загрузить работой все ядра, в программе должно быть несколько параллельных потоков

ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

- Подробности о памяти: память имеет иерархическую структуру
  - Уровни: регистры, кэши L1, L2, L3, оперативная память, дисковый своп
  - Уровни различаются по объёму и скорости доступа
  - Память оптимизирована для последовательного доступа

Ulrich Drepper, What every programmer should know about memory

Пример: время доступа к элементу данных в зависимости от объёма данных

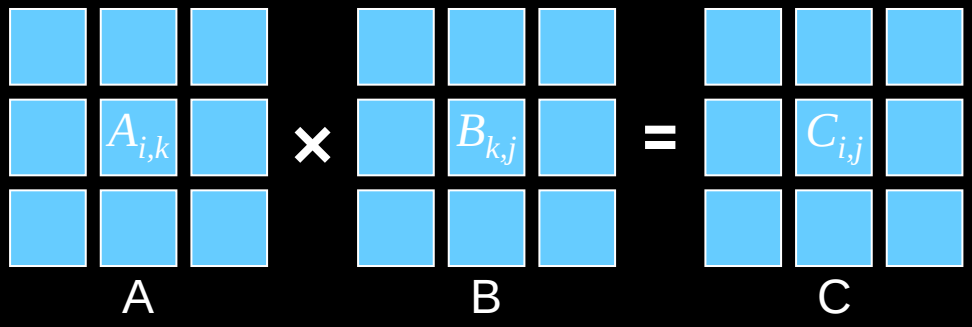


ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
  - Пример оптимизации: блочные (tiled) алгоритмы



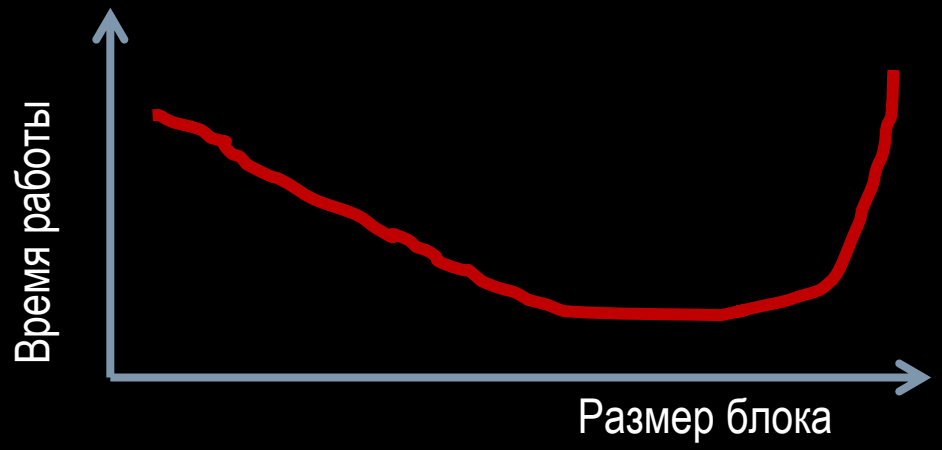
ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
    - Пример оптимизации: блочные (tiled) алгоритмы

      (оптимальный размер блока зависит от параметров компьютера)

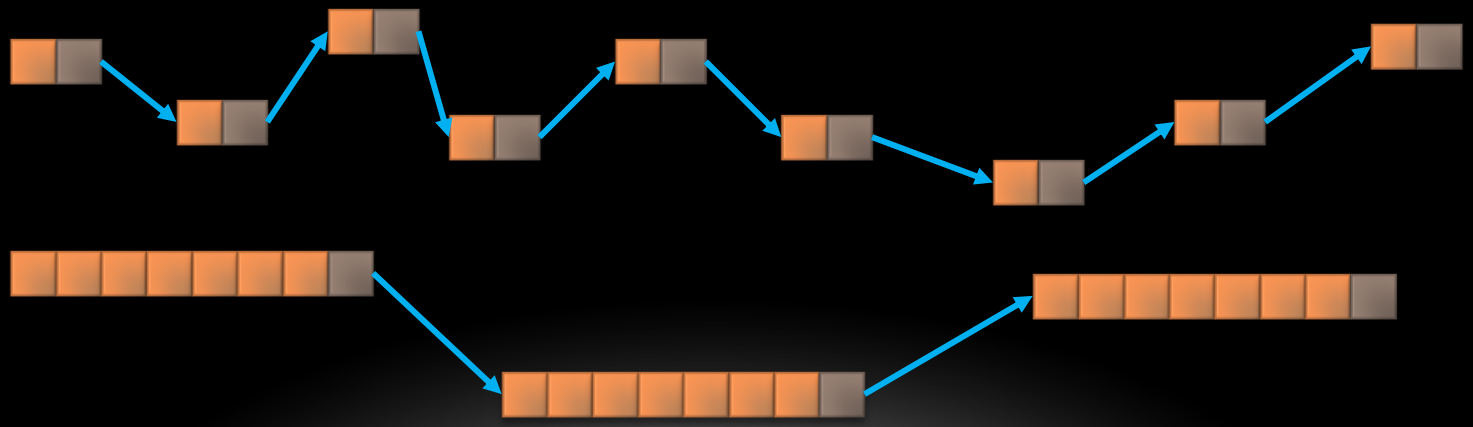


ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
- Вывод 2: данные нужно хранить плотно, без «дырок»
  - Пример оптимизации: список массивов



ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
- Вывод 2: данные нужно хранить плотно, без «дырок»
  - Пример оптимизации: список массивов вместо списка
  - Пример оптимизации: структура массивов вместо массива структур (SoA vs AoS)

```
struct {
  double x;
  int count;
} s[N];
```

```
double x[N];
int count[N];
```

Размер: 16 N байт

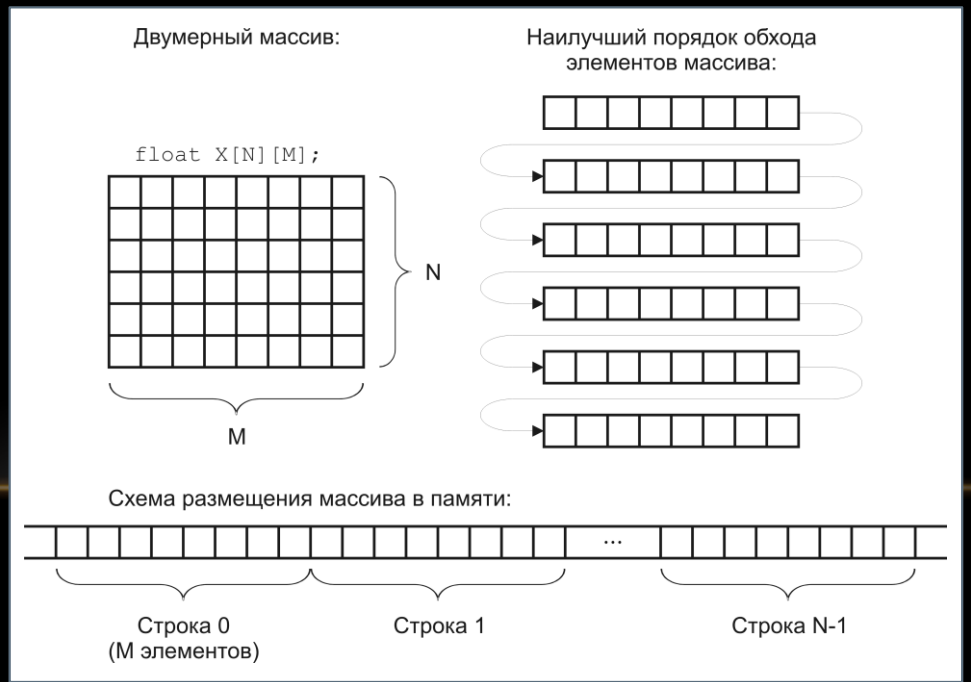
Размер: 12 N байт

ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
- Вывод 2: данные нужно хранить плотно, без «дырок»
- Вывод 3: данные нужно стараться обходить последовательно



ВТОРОЙ УРОВЕНЬ. ПАМЯТЬ

Уровни памяти различаются по объёму и скорости доступа

Память оптимизирована для последовательного доступа

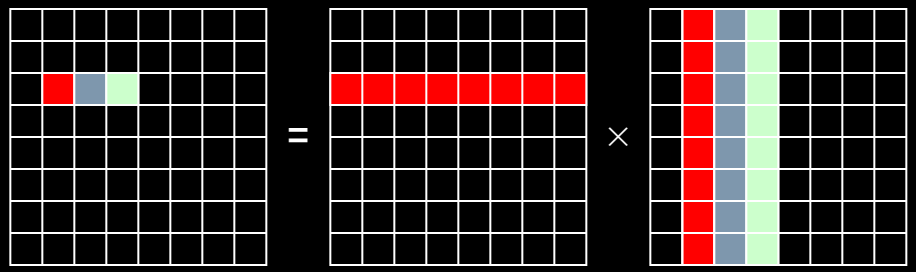
- Вывод 1: чем меньше объём активных данных программы, тем быстрее к ним доступ
- Вывод 2: данные нужно хранить плотно, без «дырок»
- Вывод 3: данные нужно стараться обходить последовательно
  - Примеры оптимизаций: перестановка циклов, перестановка размерностей массивов (транспонирование)

ВТОРОЙ УРОВЕНЬ: ПАМЯТЬ

Пример: умножение матриц

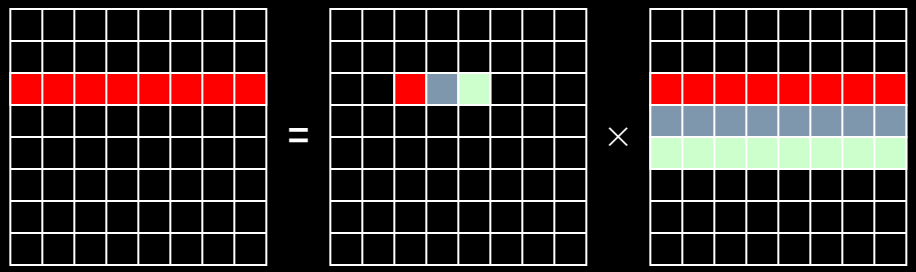
по формуле: $C_{ik} = \sum_{j=0}^{Ny-1} A_{ij} B_{jk}$

```
for(i=0;i<Nx-1;i++)
 for(k=0;k<Nz-1;k++)
  for(j=0;j<Ny-1;j++)
   C[i][k]+=A[i][j]*B[j][k];
```



переставим циклы

```
for(i=0;i<Nx-1;i++)
 for(j=0;j<Ny-1;j++)
  for(k=0;k<Nz-1;k++)
   C[i][k]+=A[i][j]*B[j][k];
```



Время перемножения матриц 2000×2000		
17.37 с	**Core i7-7700**	3.86 с

Анализ работы программы:
Профиль доступа в память

Локальность по пространству
и по времени

Пример профиля работы с памятью для задачи DIS-класса (SpMV)

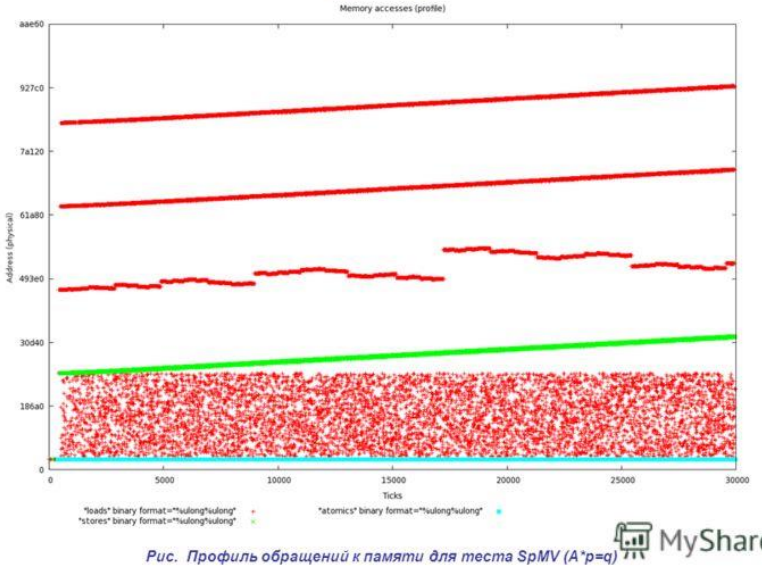
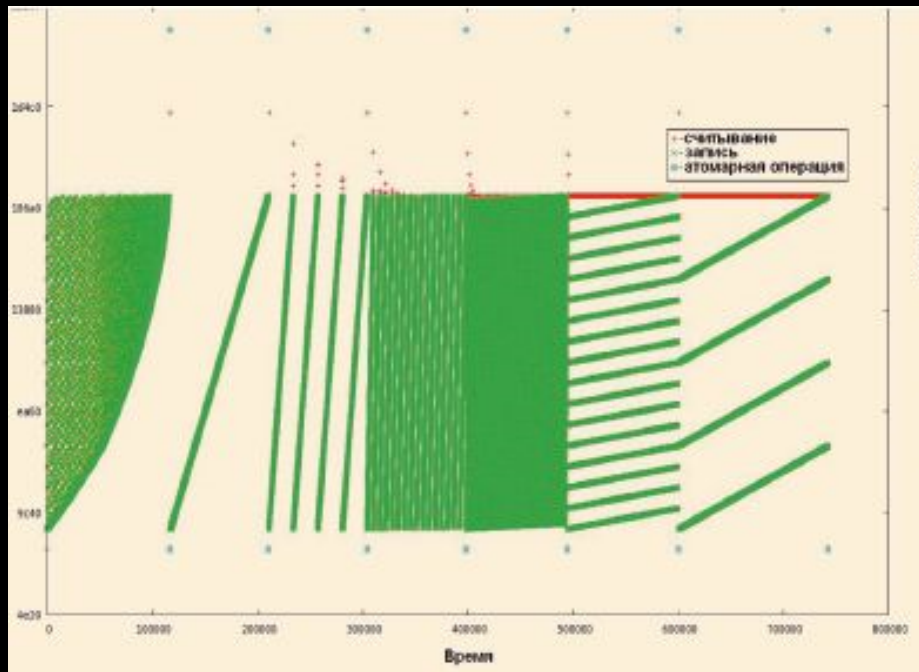
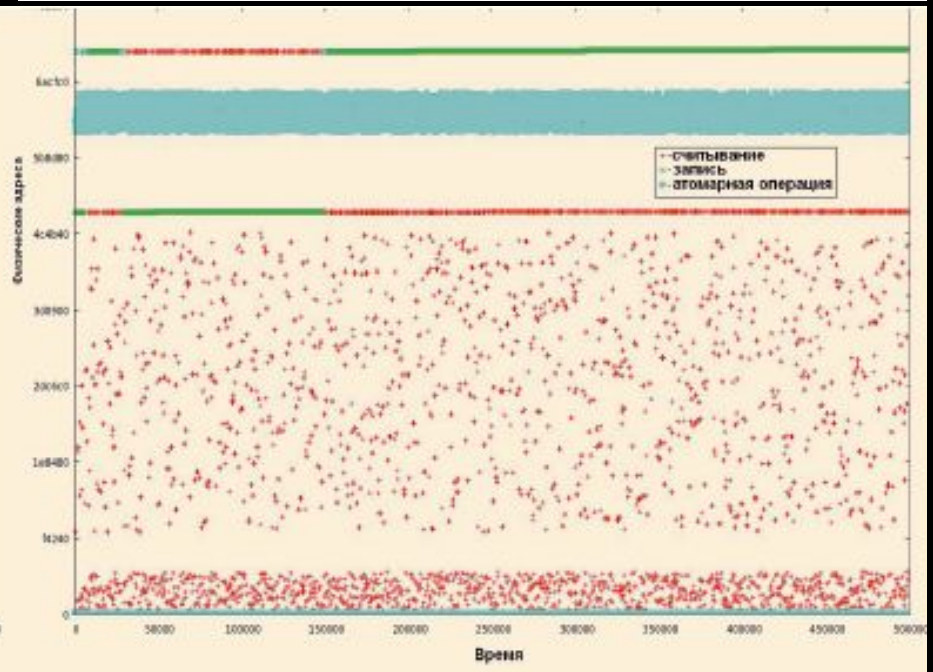


*Рис. Профиль обращений к памяти для теста SpMV (A*p=q)*



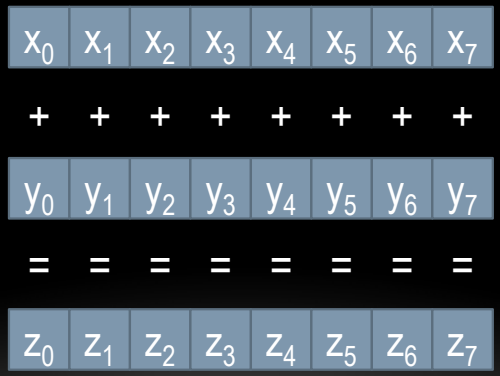
FFT - вариант нерегулярного доступа



BFS - вариант нерегулярного доступа

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

- Подробности о векторизации
  - Процессор может за одну команду выполнить операции над несколькими аргументами и получить несколько результатов
  - Разные процессоры поддерживают разные размеры векторов с разными типами данных и разные операции над ними



ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

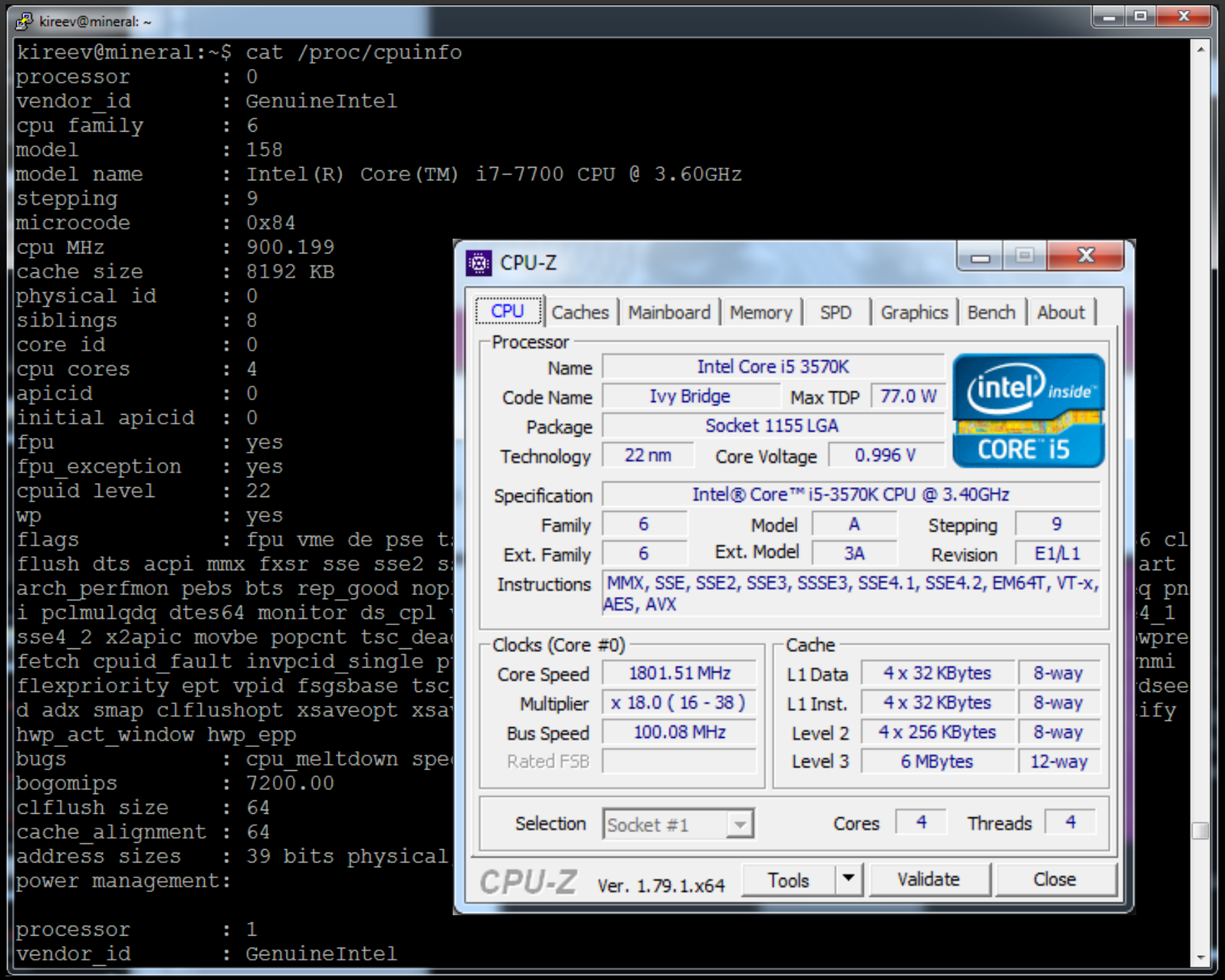
Векторные расширения архитектуры x86

- Расширение MMX, 3DNow!, MMX Extended – Pentium MMX, K6-2
  - Регистры: 8 × 8 байт
  - Типы данных: целочисленные, float
  - Базовые операции, арифметика с насыщением, двухоперандные операции: x += y
- Расширение SSE – Pentium III, Athlon XP
  - Регистры: 8 (16) × 16 байт
  - Типы данных: float
  - Базовые операции
- Расширения SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, SSE4a – Pentium 4, Opteron, …
  - Регистры: 8 (16) × 16 байт
  - Типы данных: целочисленные, float, double
  - +Горизонтальные операции, специальные операции
- Специальные расширения: CLMUL, AES, XOP, F16C, FMA4, FMA3, …
- Расширение AVX – Ivy Bridge, Bulldozer
  - Регистры: 16 × 32 байта
  - Типы данных: float, double
  - Трёхоперандные неразрушающие операции: z = x + y
  - +Отсутствуют требования к выравниванию (но это влияет на производительность)
- Расширение AVX2 – Haswell, Ryzen
  - Регистры: 16 × 32 байта
  - Типы данных: целочисленные, float, double
- Расширение AVX-512 – Xeon Phi, Skylake (Xeon)
  - Регистры: 16 × 64 байта
  - Типы данных: целочисленные, float, double

kireev@mineral: ~

```
kireev@mineral:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
stepping        : 9
microcode       : 0x84
cpu MHz         : 900.199
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
flush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pn
i pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowpre
fetch cpuid_fault invpcid_single pti retpoline intel_pt rsb_ctxsw spec_ctrl tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdsee
d adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify
hwp_act_window hwp_epp
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips        : 7200.00
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
```


kireev@mineral: ~
kireev@mineral:~$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 158
model name : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
stepping : 9
microcode : 0x84
cpu MHz : 900.199
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 0
cpu cores : 4
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 22
wp : yes
flags : fpu vme de pse t
flush dts acpi mmx fxsr sse sse2 s
arch_perfmon pebs bts rep_good nop
i pclmulqdq dtes64 monitor ds_cpl
sse4_2 x2apic movbe popcnt tsc_dea
fetch cpuid_fault invpcid_single p
flexpriority ept vpid fsgsbase tsc
d adx smap clflushopt xsaveopt xsa
hwp_act_window hwp_epp
bugs : cpu_meltdown spe
bogomips : 7200.00
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical
power management:
processor : 1
vendor_id : GenuineIntel
CPU-Z
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
Processor
Name
Intel Core i5 3570K
Code Name
Ivy Bridge
Max TDP
77.0 W
Package
Socket 1155 LGA
Technology
22 nm
Core Voltage
0.996 V
Specification
Intel® Core™ i5-3570K CPU @ 3.40GHz
Family
6
Model
A
Stepping
9
Ext. Family
6
Ext. Model
3A
Revision
E1/L1
Instructions
MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX
Clocks (Core #0)
Core Speed
1801.51 MHz
Multiplier
x 18.0 (16 - 38)
Bus Speed
100.08 MHz
Rated FSB
Cache
L1 Data
4 x 32 KBytes
8-way
L1 Inst.
4 x 32 KBytes
8-way
Level 2
4 x 256 KBytes
8-way
Level 3
6 MBytes
12-way
Selection
Socket #1
Cores
4
Threads
4
CPU-Z
Ver. 1.79.1.x64
Tools
Validate
Close

СКАЛЯРНЫЕ ИЛИ ВЕКТОРНЫЕ, В ЧЁМ РАЗНИЦА?

• Скалярные инструкции:			1 × double	
• addsd	a + b	latency: 3, 4	throughput: 1, 0.8	
• mulsd	a * b	latency: 3, 5	throughput: 1, 0.5	
• divsd	a / b	latency: < 14 - 22	throughput: 4 - 22	
• Векторные SSE2:			2 × double	
• addpd	a + b	latency: 3, 4	throughput: 1, 0.8	
• mulpd	a * b	latency: 3, 5	throughput: 1, 0.5	× 2
• divpd	a / b	latency: < 14 - 22	throughput: 4 - 22	
• Векторные AVX:			4 × double	
• vaddpd	a + b	latency: 3, 4	throughput: 1	
• vmulpd	a * b	latency: 3, 4, 5	throughput: 1, 0.5	× 4
• vdivpd	a / b	latency: 14 - 35	throughput: 8 - 28	
• Векторные AVX2:			4 × double	
• vfmadd132pd	a * b + c	latency: 5	throughput: 0.5	× 4+

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

- Что делать?
  1. Найти цикл, заметно влияющий на время работы программы
  2. Определить, какой тип данных используется в цикле, какие с ним выполняются операции
  3. Выбрать наилучшее подходящее векторное расширение из поддерживаемых процессором
  4. Векторизовать цикл с использованием этого векторного расширения

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

- Как векторизовать?
  - Автоматически
  - Полуавтоматически
  - Вручную

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

Автоматическая векторизация

- Компилятор распознаёт циклы, которые могут быть векторизованы, и векторизует их.
- Пользователь может сообщать компилятору дополнительную информацию и пожелания с помощью директив.
- Где работает:
  - **gcc** (циклы попроще), **icc** (циклы посложнее)
- Пример: скалярное произведение векторов длины n:

```
float inner(int n, float* x, float* y) {
  float s = 0.0f;
  for(int i=0; i<n; ++i)
    s += x[i] * y[i];
  return s;
}
$icc -vec-report=3 test.c
…
test.c(21): (col. 3) remark: LOOP WAS VECTORIZED
…
```

A Guide to Auto-vectorization with Intel® C++ Compilers

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

Полуавтоматическая векторизация

- Для задания правил векторизации используются директивы OpenMP 4.0
- Программист сам должен следить за корректностью применения директив

Где работает:

- Компиляторы, поддерживающие OpenMP 4.0
  - icc, gcc-4.9 -fopenmp-simd
- Пример: скалярное произведение векторов длины n:

```
float inner(int n, float* x, float* y){
  float s = 0.0f;
  #pragma omp simd reduction(+:s)
  for(int i=0; i<n; ++i)
    s += x[i] * y[i];
  return s;
}
```

Intel SIMD vectorization: https://software.intel.com/en-us/node/522575
OpenMP 4.0: http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf

ВТОРОЙ УРОВЕНЬ. ВЕКТОРИЗАЦИЯ

Ручная векторизация

Используются встроенные в компилятор операции (intrinsics) и типы данных:

- Для каждого представления векторного регистра есть свой тип данных: __m128, __m128d, __m128i
- Для каждой векторной команды процессора есть своя встроенная функция: _mm256_add_pd

Где работает:

- На большинстве известных компиляторов (gcc, clang, icc, cl.exe, …)
- Компилятор должен поддерживать используемое векторное расширение

Пример: скалярное произведение векторов длины n, кратной 4-м, с использованием расширения SSE

```
#include <xmmintrin.h>
float inner(int n, float* x, float* y){
  __m128 *xx = (__m128*)x;
  __m128 *yy = (__m128*)y;
  __m128 s = _mm_setzero_ps();
  for (int i=0; i<n/4; ++i) {
    __m128 p = _mm_mul_ps(xx[i],yy[i]);
    s = _mm_add_ps(s,p);
  }
  __m128 p = _mm_movehl_ps(p,s);
  s = _mm_add_ps(s,p);
  p = _mm_shuffle_ps(s,s,1);
  s = _mm_add_ss(s,p);
  float sum;
  _mm_store_ss(&sum,s);
  return sum;
}
```

Intel Intrinsics Guide

ВТОРОЙ УРОВЕНЬ: РАСПАРАЛЛЕЛИВАНИЕ

- Цель – занять полезной работой все ядра

- Какие части программы могут быть распараллелены?
- Какие средства многопоточного программирования существуют?

ВТОРОЙ УРОВЕНЬ: РАСПАРАЛЛЕЛИВАНИЕ

- Какие части программы могут быть распараллелены?
  - Занимающие много времени,
  - Которые могут быть разделены на независимые части
- Например:
  - Независимые итерации циклов
  - Вызовы «чистых» функций (без побочных эффектов)

ВТОРОЙ УРОВЕНЬ: РАСПАРАЛЛЕЛИВАНИЕ

- Какие средства многопоточного программирования существуют?
  - Средства автоматического распараллеливания: компиляторы
    - простые циклы без зависимостей
  - Средства «высокого» уровня: OpenMP, Cilk Plus (C/C++/Fortran)
    - явное выделение параллельных частей программы, распараллеливание циклов, параллелизм задач
  - Средства «среднего» уровня: TBB – Threading Building Blocks (C++)
    - шаблоны многопоточного программирования (очереди задач, динамическая декомпозиция, …), lock-free контейнеры
  - Средства «низкого» уровня – библиотеки потоков: POSIX Threads, WinAPI, Qt threads, C++ threads
    - ручное управление потоками: явное создание, удаление и синхронизация базовыми средствами

ВТОРОЙ УРОВЕНЬ

- Пример: параллелизм задач

OpenMP 4.0	Cilk Plus

`int fib(int n) {`
`  if (n<=2) return 1;`
`  #pragma omp task`
`  int n1 = fib(n-2);`
`  int n2 = fib(n-1);`
`  #pragma omp taskwait`
`  return n1+n2;`
`}`
`...`
`#pragma omp parallel`
`#pragma omp single`
`m = fib(n);`
	`int fib(int n) {`
	`  if (n<=2) return 1;`
	`  int n1 = cilk_spawn fib(n-2);`
	`  int n2 = fib(n-1);`
	`  cilk_sync;`
	`  return n1+n2;`
	`}`
	
	`...`
	`m = fib(n);`
	`...`

ПЛАН ДОКЛАДА

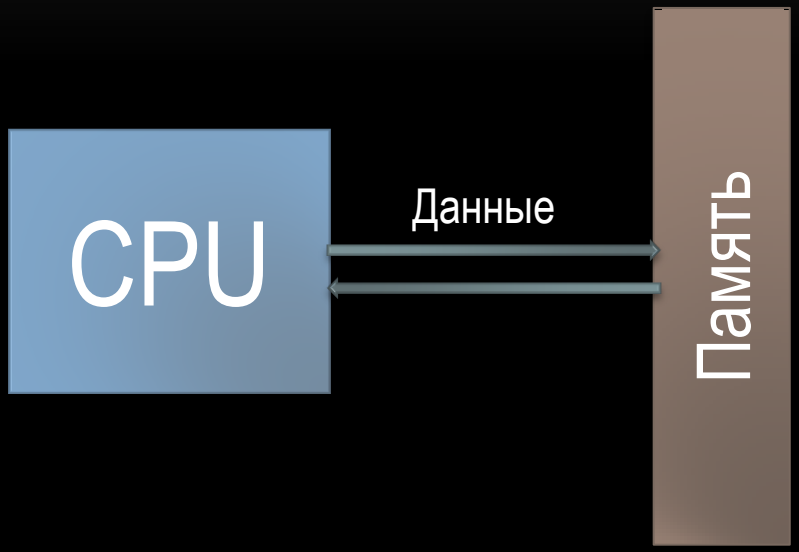
- Первый уровень вникания в проблему
  - Компилятор + библиотеки
- Второй уровень вникания в проблему:
  - Память + векторизация + распараллеливание
- Третий уровень вникания в проблему

ТРЕТИЙ УРОВЕНЬ

- Принцип: оптимизируем по максимуму

- Что нужно знать на третьем уровне?
  - Предел возможностей своего оборудования
  - Причины, по которым предел не достигнут
  - Способы их преодоления

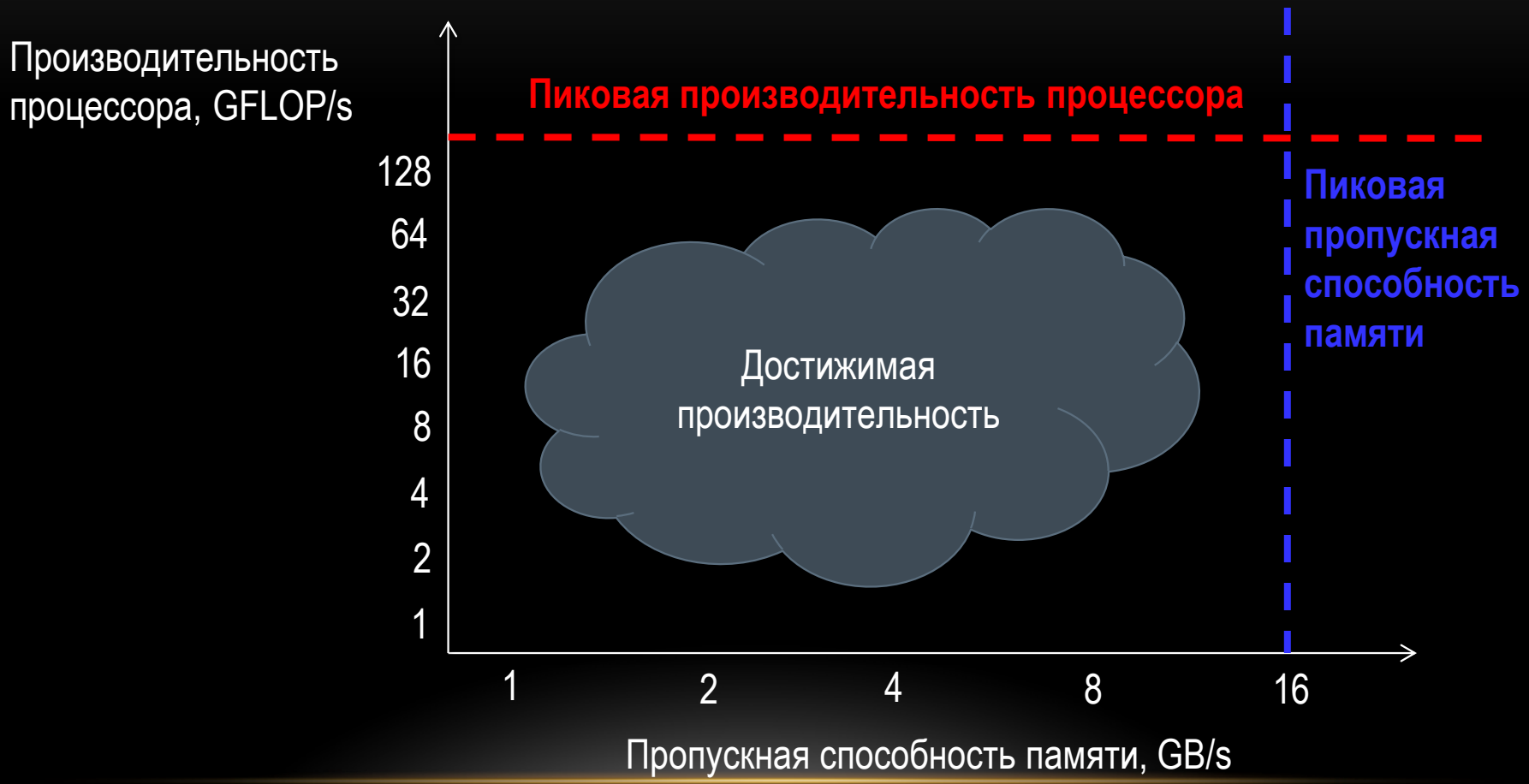
ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ



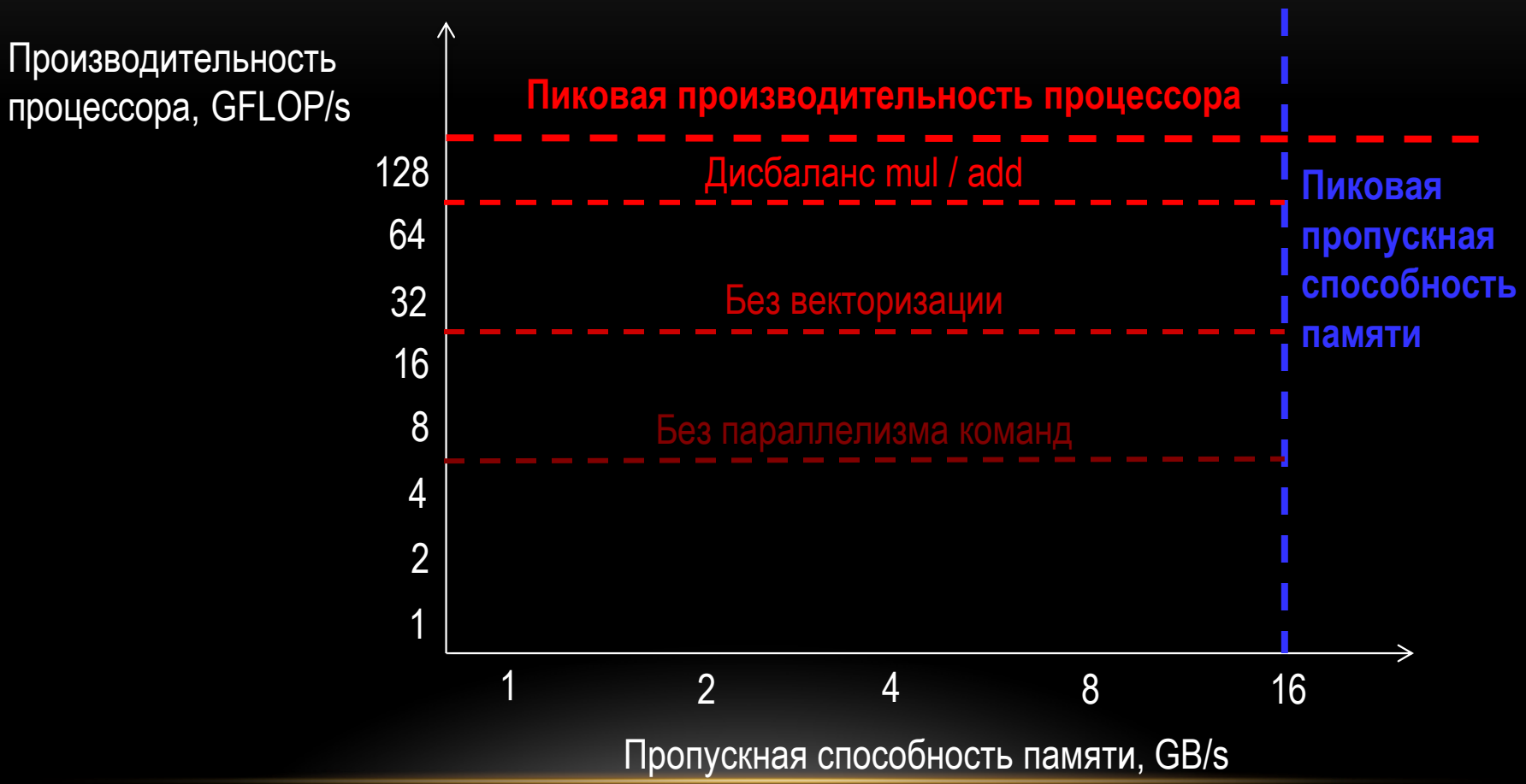
Характеристики производительности

	Процессор	Память
Время	Время выполнения операции, ns	Время доступа к элементу данных, ns
Пропускная способность	Число операций в единицу времени, Op/s	Число элементов данных в единицу времени, B/s

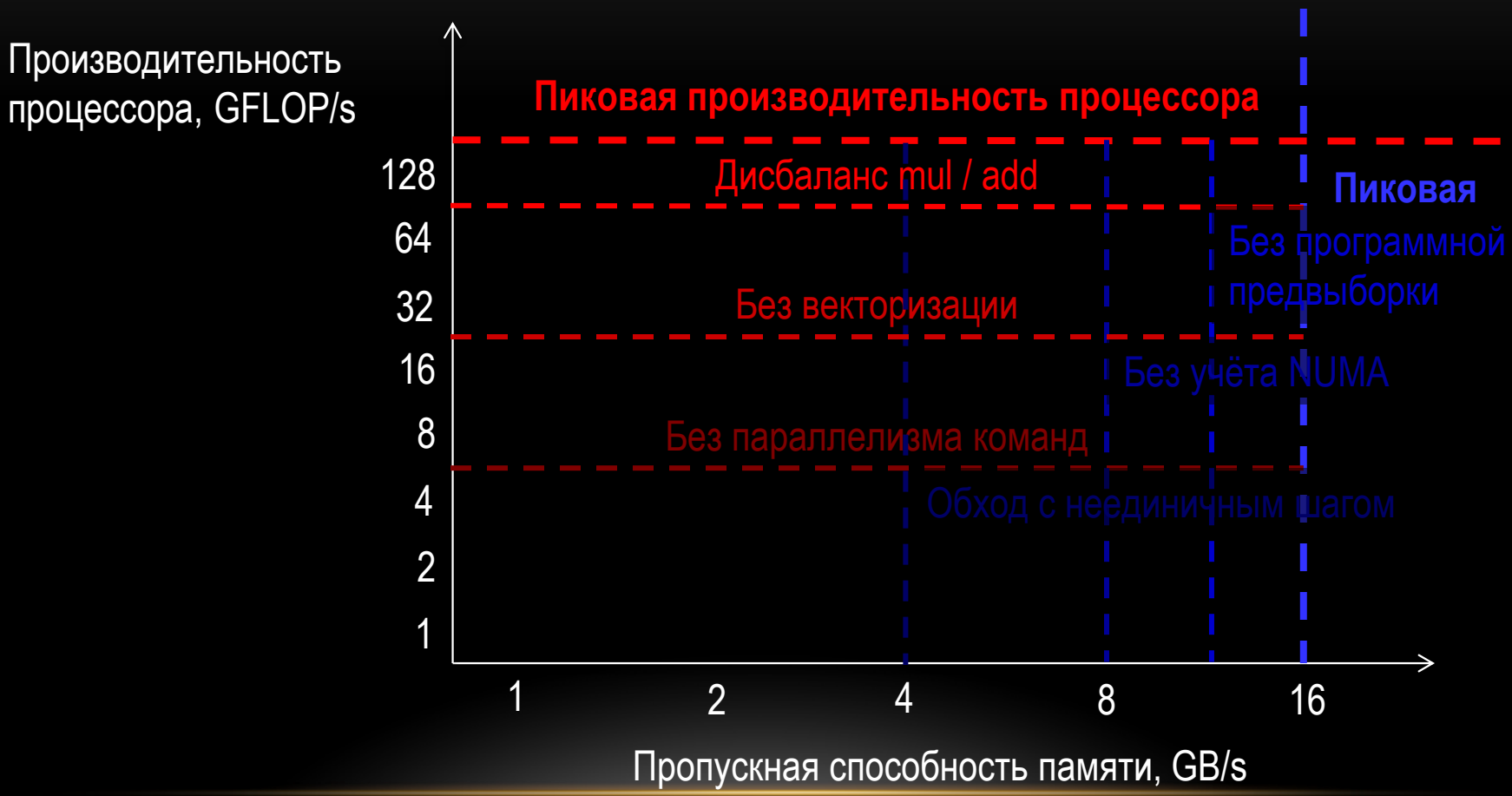
ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ



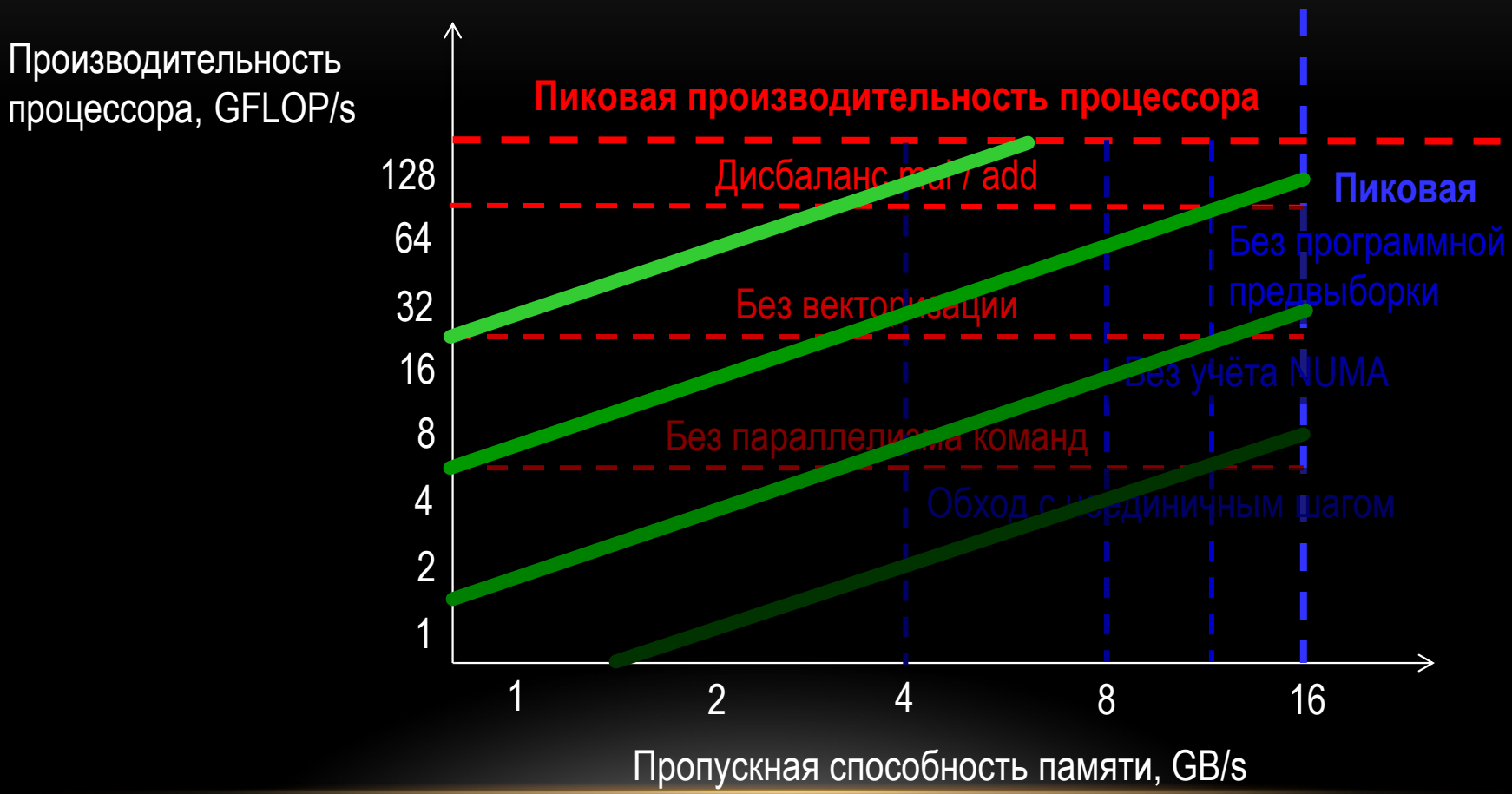
ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ



ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ



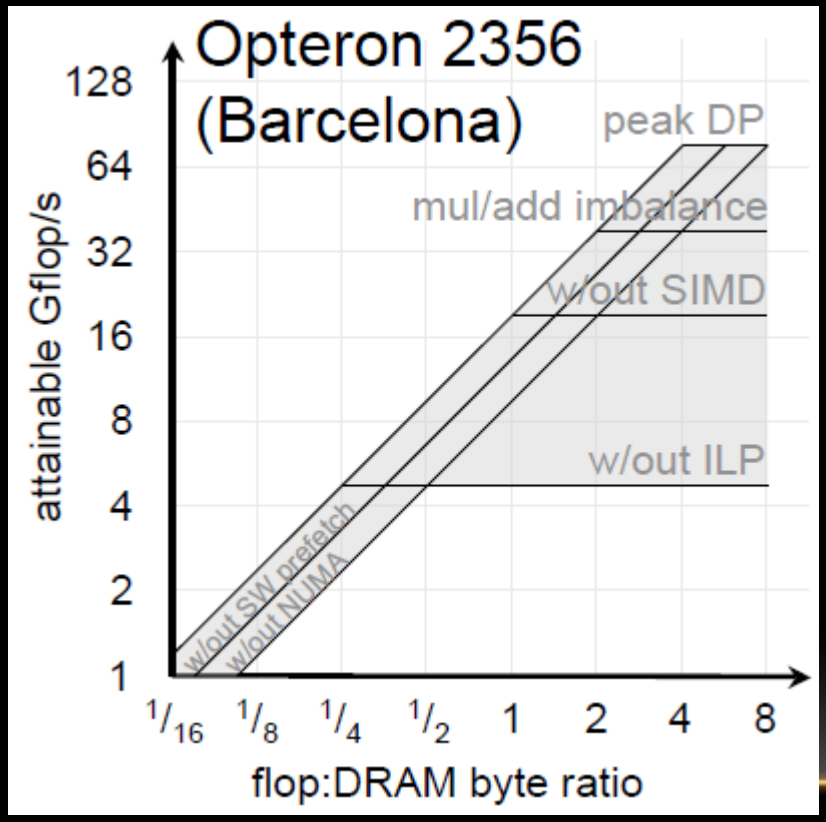
ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ

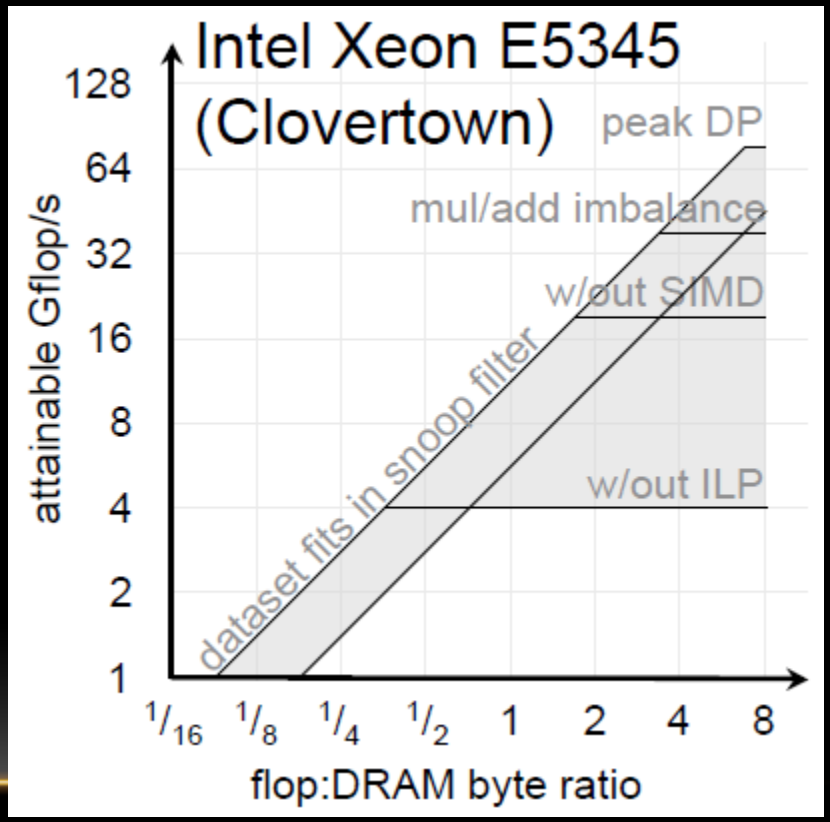


Характеристика программы: Интенсивность вычислений, FLOP/B

ТРЕТИЙ УРОВЕНЬ. ПРЕДЕЛ ОБОРУДОВАНИЯ

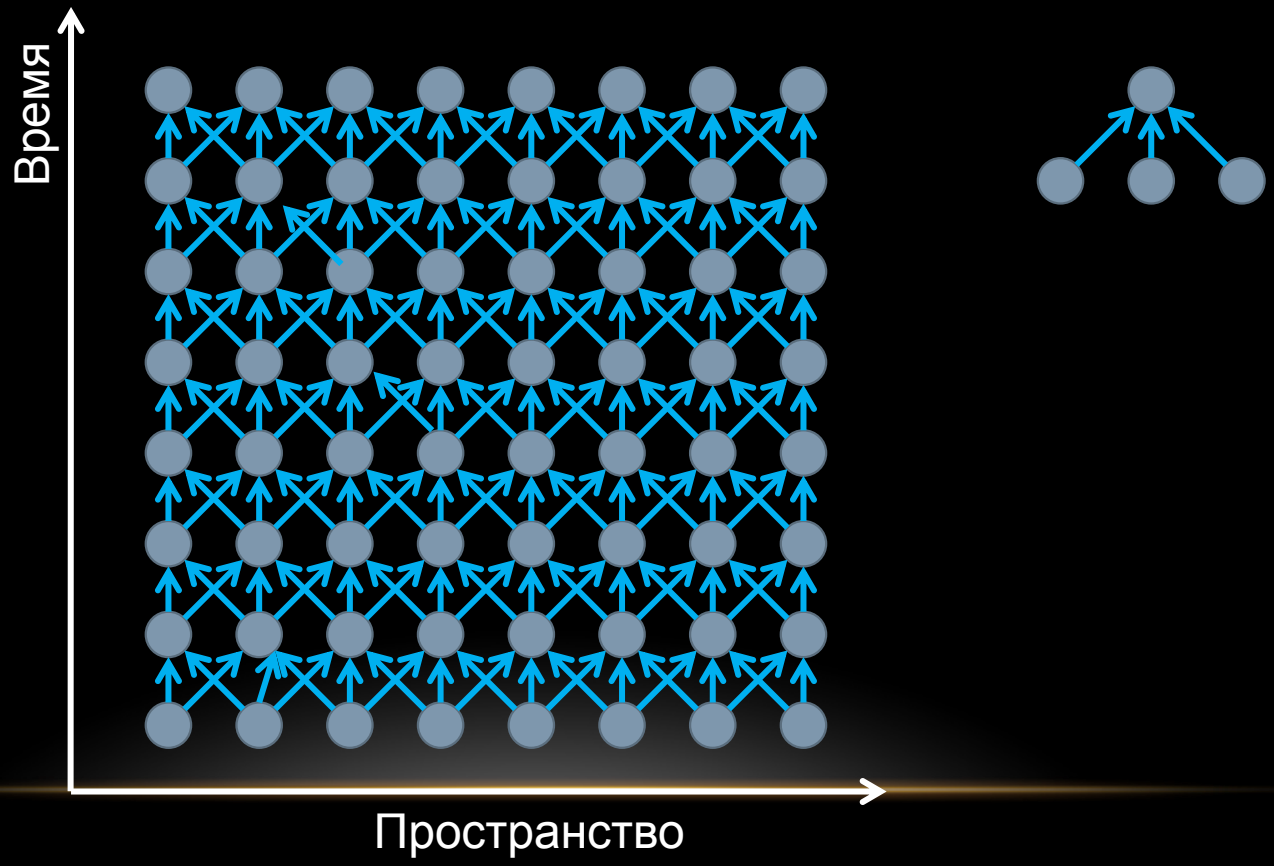
Roofline Performance Model





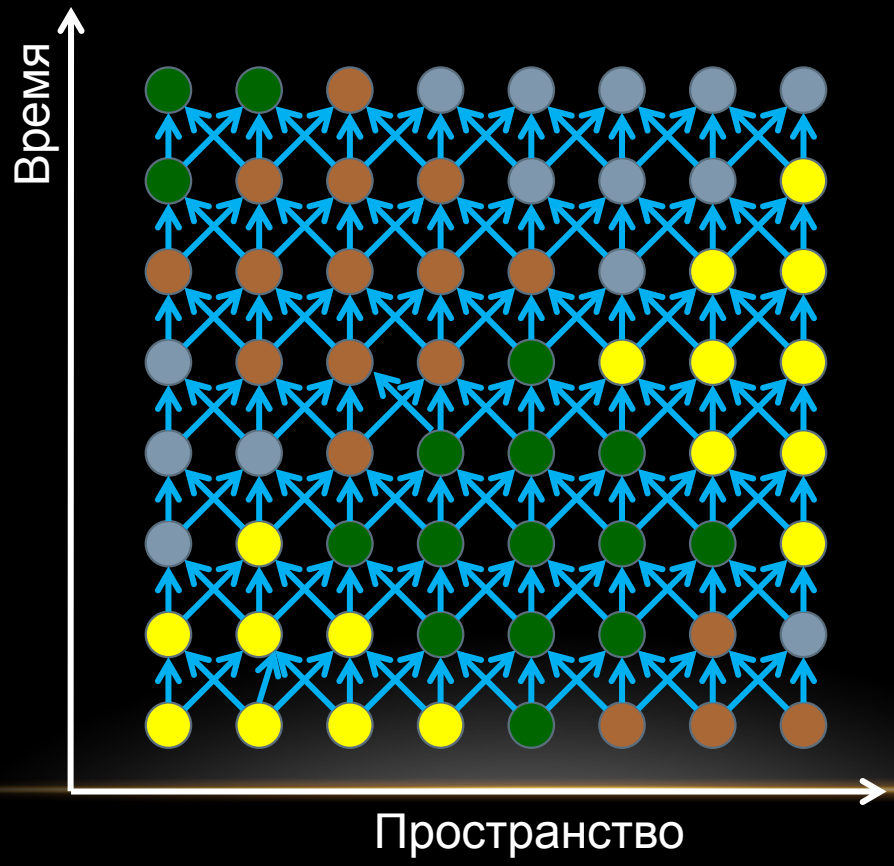
ТРЕТИЙ УРОВЕНЬ

- Пример шага за край: трафаретные вычисления (1D)



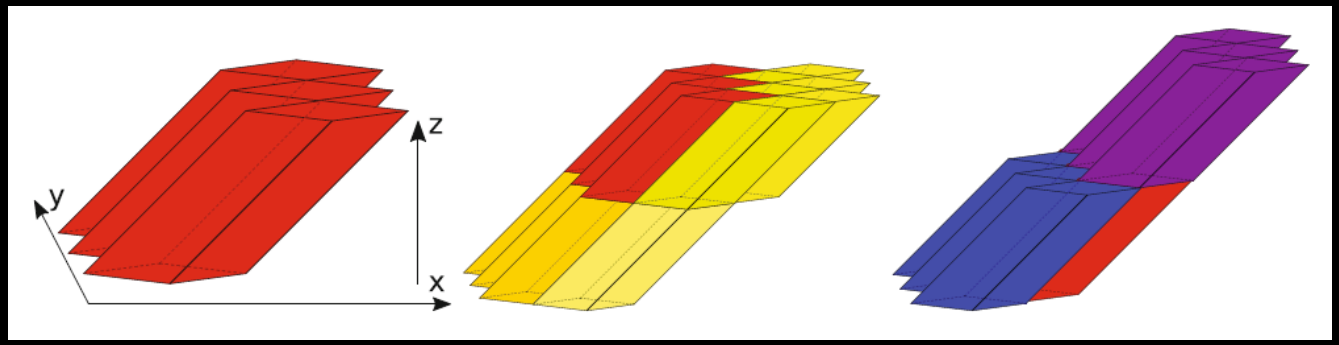
ТРЕТИЙ УРОВЕНЬ

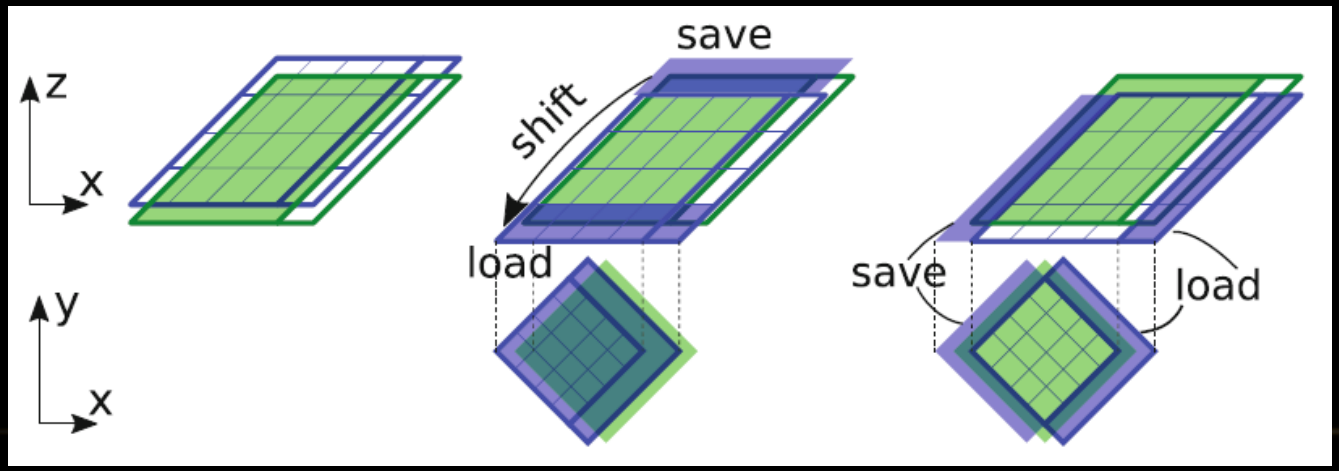
- Пример шага за край: трафаретные вычисления (1D)



ТРЕТИЙ УРОВЕНЬ

- Пример шага за край: трафаретные вычисления (3D)

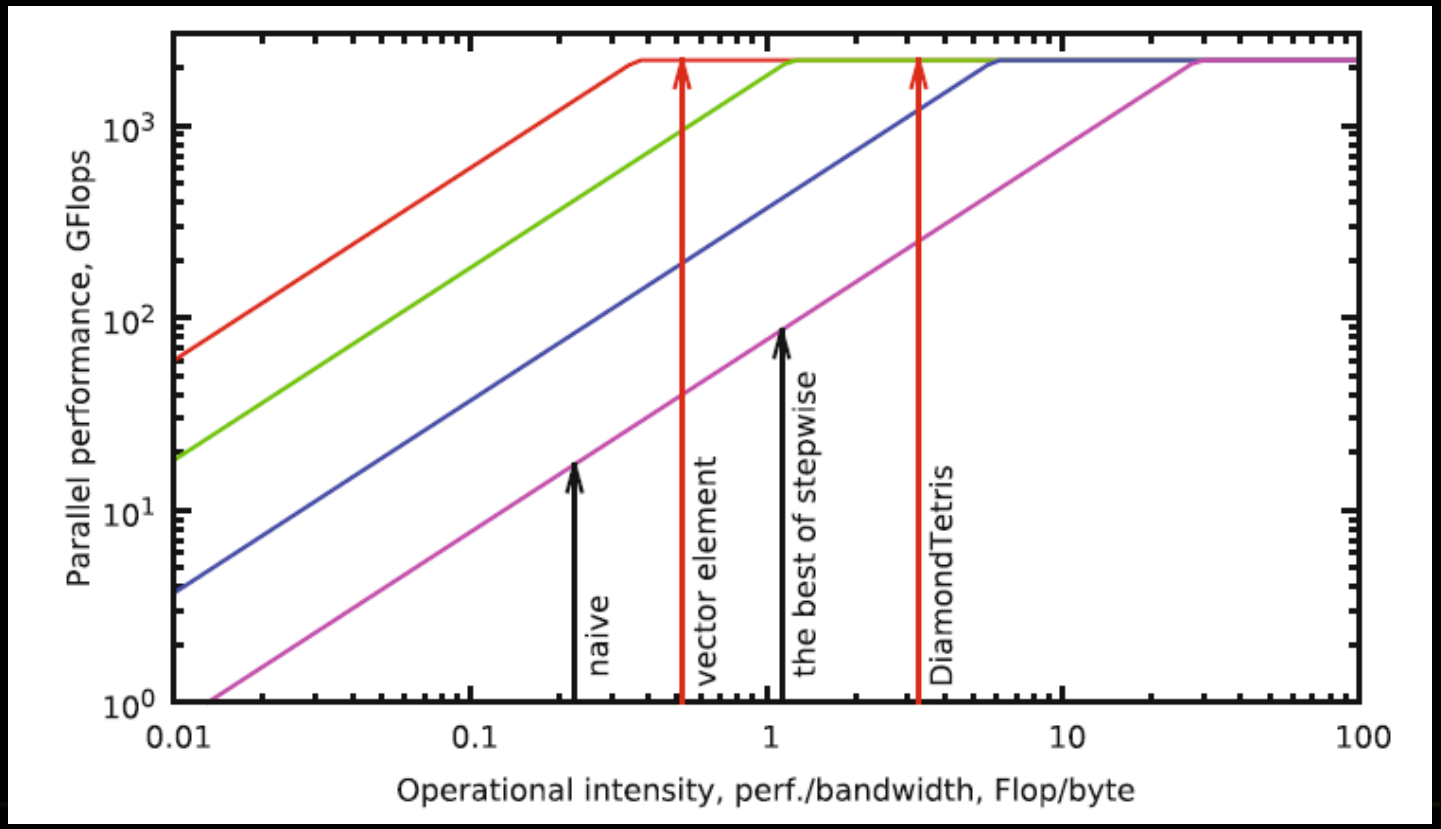




Vadim Levchenko and Anastasia Perepelkina «**The DiamondTetris Algorithm for Maximum Performance Vectorized Stencil Computation**» // PaCT 2017, LNCS 10421, pp. 124–135, 2017.

ТРЕТИЙ УРОВЕНЬ

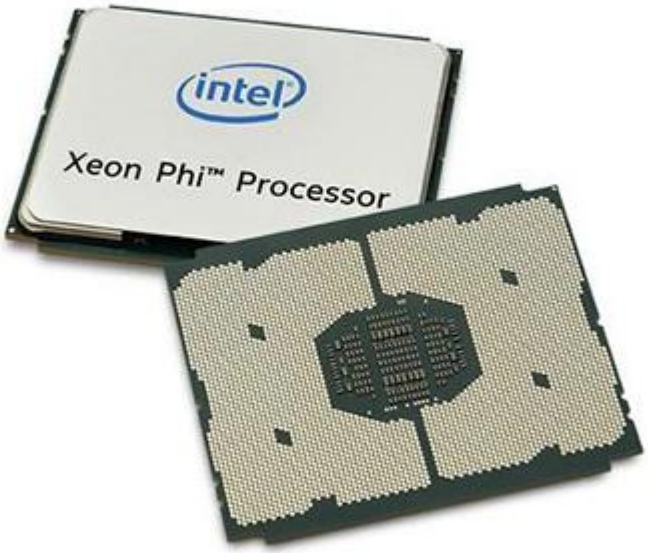
- Пример шага за край: трафаретные вычисления (3D)



Vadim Levchenko and Anastasia Perepelkina «**The DiamondTetris Algorithm for Maximum Performance Vectorized Stencil Computation**» // PaCT 2017, LNCS 10421, pp. 124–135, 2017.

ИСТОЧНИКИ

- Спецкурс «Эффективное программирование современных микропроцессоров и мультипроцессоров», ФИТ НГУ
  http://ssd.sscc.ru/ru/chair/nsu/programming


intel
Xeon Phi™ Processor