

# Лабораторная работа 1

## Оптимизация доступа к памяти, циклов и ветвлений

**Курносов Михаил Георгиевич**

E-mail: [mkurnosov@gmail.com](mailto:mkurnosov@gmail.com)

WWW: [www.mkurnosov.net](http://www.mkurnosov.net)

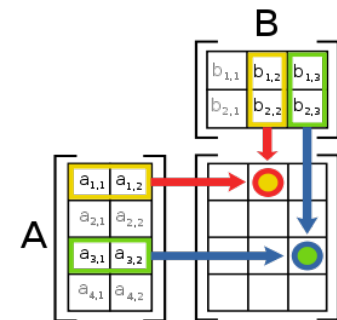
Курс «Параллельные вычислительные технологии»

Сибирский государственный университет телекоммуникаций и информатики (Новосибирск)

# Описание лабораторной работы

- В данной лабораторной работе рассматриваются приёмы эффективной работы с кэш-памятью процессора
- В качестве тестовой программы используется функция DGEMM умножения матриц (double-precision general matrix multiply)

```
/* DGEMM */  
for (i = 0; i < N; i++) {  
    for (j = 0; j < N; j++) {  
        for (k = 0; k < N; k++) {  
            c[i][j] += a[i][k] * b[k][j];  
        }  
    }  
}
```



$$c_{ij} = \sum_{k=1}^N a_{ik} b_{kj}$$

# Задание 1

---

- В файле **dgemm.c** реализовать следующие функции:

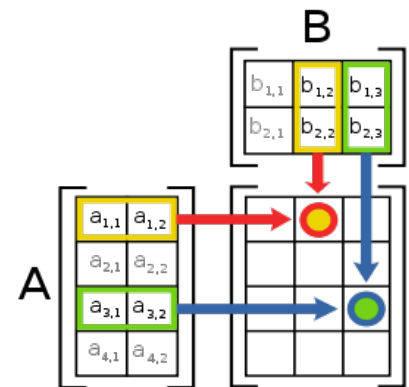
- ☐ **dgemm\_transpose**

Версия с переставленными местами циклами:  $ijk \rightarrow ikj$

- ☐ **dgemm\_block**

Блочный алгоритм умножения матриц

```
/* DGEMM */
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        for (k = 0; k < N; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```



# Задание 1

---

- Занести в таблицу время выполнения функций `dgemm_def`, `dgemm_transpose`, `dgemm_block` и оценить достигнутое ускорение

dgemm_def	dgemm_transpose		dgemm_block				
Time, s	Time, s	Speedup	BS = 2		BS = 4		...
			Time, s	Speedup	Time, s	Speedup	

- Для блочного алгоритма определить значение параметра BS при котором достигается наибольшее ускорение
- Чем объясняется достигнутое ускорение?
- На что влияет значение параметра BS?

# Задание 1

---

- Выполнить профилирование программы пакетом **Valgrind**  
<http://www.mkurnosov.net/teaching/index.php/HPC/LinuxProfilers>
- Используя Valgrind (cachegrind) сформировать аннотированный исходный текст программы и оценить число промахов по кеш-памяти данных и инструкций, которые произошли во внутреннем цикле функций `dgemm_def`, `dgemm_transpose`
- При помощи профилировщика **perf** получить аннотированный отчет с оценкой числа промахов при обращении к кеш-памяти в функциях `dgemm_def`, `dgemm_transpose`

```
$ perf record -e cache-misses ./dgemm
```

## Задание 2

---

- В программе **branch** оптимизировать функцию **blend\_map** – минимизировать количество ошибок предсказания переходов (branch misprediction)
- Провести замеры времени выполнения программ, результаты оформить в виде таблицы:

$n$	Время выполнения функции <b>blend_map</b>	Время выполнения оптимизированной функции <b>blend_map</b>	Ускорение (Speedup)
100 000			
1 000 000			

$n$  – количество элементов в массивах  $x$ ,  $y$ ,  $z$

## Задание 2

---

- Используя профилировщик **perf** оценить количество ветвлений в исходной функции **bland\_map** и её оптимизированной версии (представить аннотированный исходный код)
- Объяснить причину достигнутого ускорения

- Заполнить таблицу:

[illegible]



## Задание 3

---

- Определить профилировщиком **perf** количество ветвлений в исходной программе и в программе с развернутым циклом (представить аннотированный исходный код)
- Объяснить причину достигнутого ускорения

# Профилировщик Perf

- <https://perf.wiki.kernel.org>
- <http://www.brendangregg.com/perf.html>

# Профилировщик Perf

---

- Получение списка поддерживаемых счетчиков производительности

```
$ perf list
```

- Получение суммарной информации о значениях счетчиков производительности при выполнении программы

```
$ perf stat ./branch
```

```
Performance counter stats for './branch':
```

9.929773 task-clock	#	0.003 CPUs utilized	
5 context-switches	#	0.504 K/sec	
0 CPU-migrations	#	0.000 K/sec	
669 page-faults	#	0.067 M/sec	
22,507,821 cycles	#	2.267 GHz	[82.44%]
8,375,398 stalled-cycles-frontend	#	37.21% frontend cycles idle	[80.05%]
3,536,990 stalled-cycles-backend	#	15.71% backend cycles idle	[61.70%]
45,356,331 instructions	#	2.02 insns per cycle	
	#	0.18 stalled cycles per insn	[81.60%]
8,054,207 branches	#	811.117 M/sec	[89.54%]
15,227 branch-misses	#	0.19% of all branches	[89.99%]

```
3.010672196 seconds time elapsed
```

# Профилировщик Perf

---

- Получение информации о значениях заданного счетчика производительности

```
$ perf stat -e branch-misses ./branch
```

```
Performance counter stats for './branch':
```

```
12,388 branch-misses
```

```
3.019153892 seconds time elapsed
```

# Профилировщик Perf

---

- Формирование аннотированного исходного кода программы


```
$ perf record -e branch-misses ./branch
```

```
[ perf record: Woken up 1 times to write data ]  
[ perf record: Captured and wrote 0.018 MB perf.data (~789 samples) ]
```

# Профилировщик Perf

- Формирование аннотированного исходного кода программы  
**\$ perf report**

```
Samples: 24 of event 'branch-misses', Event count (approx.): 20860
47.72% branch [kernel.kallsyms] [k] path_openat
21.14% branch [kernel.kallsyms] [k] perf_event_mmap
11.06% branch [kernel.kallsyms] [k] ttwu_stat
8.42% branch [kernel.kallsyms] [k] free_pgd_range
4.61% branch libc-2.15.so [.] _IO_file_xsputn@@GLIBC_2.2.5
2.19% branch [kernel.kallsyms] [k] _raw_spin_lock
1.02% branch [kernel.kallsyms] [k] perf_event_task_tick
0.85% branch libc-2.15.so [.] vfprintf
0.85% branch [kernel.kallsyms] [k] vfs_read
0.46% branch [kernel.kallsyms] [k] handle_edge_irq
0.41% branch [kernel.kallsyms] [k] put_prev_task_fair
0.33% branch [kernel.kallsyms] [k] update_curr
0.33% branch libc-2.15.so [.] __mpn_mul_1
0.30% branch [kernel.kallsyms] [k] page_add_file_rmap
0.14% branch [vboxdrv] [k] 0x000000000000f1a1
0.09% branch [kernel.kallsyms] [k] perf_pmu_rotate_start.isra.52
0.06% branch [kernel.kallsyms] [k] pipe_wait
0.02% branch [kernel.kallsyms] [k] hrtimer_interrupt
0.01% branch branch [.] blend_map
```



Press '?' for help on key bindings

# Профилировщик Perf

- Формирование аннотированного исходного кода программы  
**\$ perf report**

```
blend_map
double x[n], y[n], z[n];
void blend_map(double *dest, double *a, double *b, int size, int blend)
{
    push    %rbp
    mov     %rsp,%rbp
    mov     %rdi,-0x18(%rbp)
    mov     %rsi,-0x20(%rbp)
    mov     %rdx,-0x28(%rbp)
    mov     %ecx,-0x2c(%rbp)
    mov     %r8d,-0x30(%rbp)
    int i = 0;
    movl    $0x0,-0x4(%rbp)

    for (i = 0; i < size; i++) {
        movl    $0x0,-0x4(%rbp)
        ↓ jmpq    111
        2a:      if (blend == 255) {
            cmpl    $0xff,-0x30(%rbp)
            ↓ jne     66
                dest[i] = a[i];
            mov     -0x4(%rbp),%eax
            cltq
            lea     0x0(,%rax,8),%rdx
            mov     -0x18(%rbp),%rax
            add     %rax,%rdx
            mov     -0x4(%rbp),%eax
            cltq
            lea     0x0(,%rax,8),%rcx
            ↓ jmpq    111
        }
    }
}
Press 'h' for help on key bindings
```