Векторизация циклов в открытых компиляторах для архитектур с короткими векторными регистрами

Ольга Владимировна Молдованова [1,2] **Иван Иванович Кулагин** [1,2] **Михаил Георгиевич Курносов** [1,2]

ovm@sibguti.ru, ovm@isp.nsc.ru ivan.i.kulagin@gmail.com WWW: www.mkurnosov.net

[1] Кафедра вычислительных систем, Сибирский государственный университет телекоммуникаций и информатики, Новосибирск

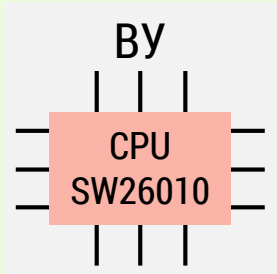
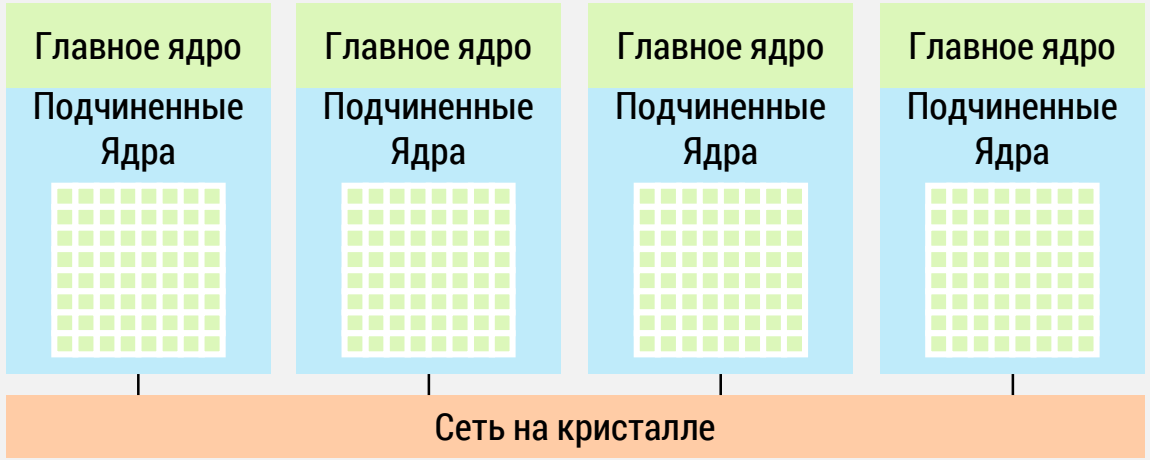
[2] Лаборатория вычислительных систем, Институт физики полупроводников им. А.В. Ржанова СО РАН, Новосибирск

Тринадцатая международная азиатская школа-семинар «Проблемы оптимизации сложных систем» (ПОСС-2017)

в рамках международной мультиконференции IEEE SIBIRCON 2017

18-22 сентября 2017 г., Академгородок, Новосибирск, Россия

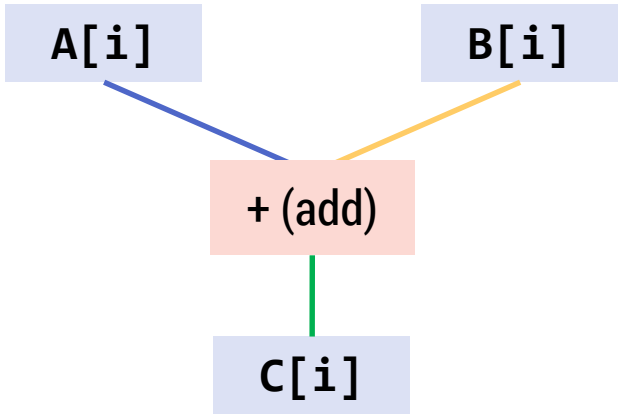
Вычислительные системы с иерархическим параллелизмом

	Sunway TaihuLight (№ 1 TOP500)	**Titan** (№ 4 TOP500)
Параллелизм уровня вычислительных узлов	40 960 × 	18 688 × (1 – AMD Opteron 6274) (1 – NVIDIA Tesla K20)
Параллелизм потоков на уровне ядер	1 ВУ = 1 CPU (SW26010) = 260 ядер = (4 × [64 + 1]) ядер 	1 ВУ = 16 ядер CPU AMD Opteron 6274 + 2 496 ядра NVIDIA Tesla K20 (SIMT – Single Instruction Multiple Threads)
Параллелизм команд на уровне АЛУ	Процессор на базе архитектуры **Alpha / SPARC ?** Ядро микроархитектуры **ShenWei** (8 АЛУ)	AMD Opteron 6274: Микроархитектура **Buildozer** (8 АЛУ)
Параллелизм уровня данных	Подчиненные ядра SW26010: поддержка векторных SIMD-команд	Ядра AMD Opteron 6274: поддержка векторного расширения **AVX**

Векторные операции

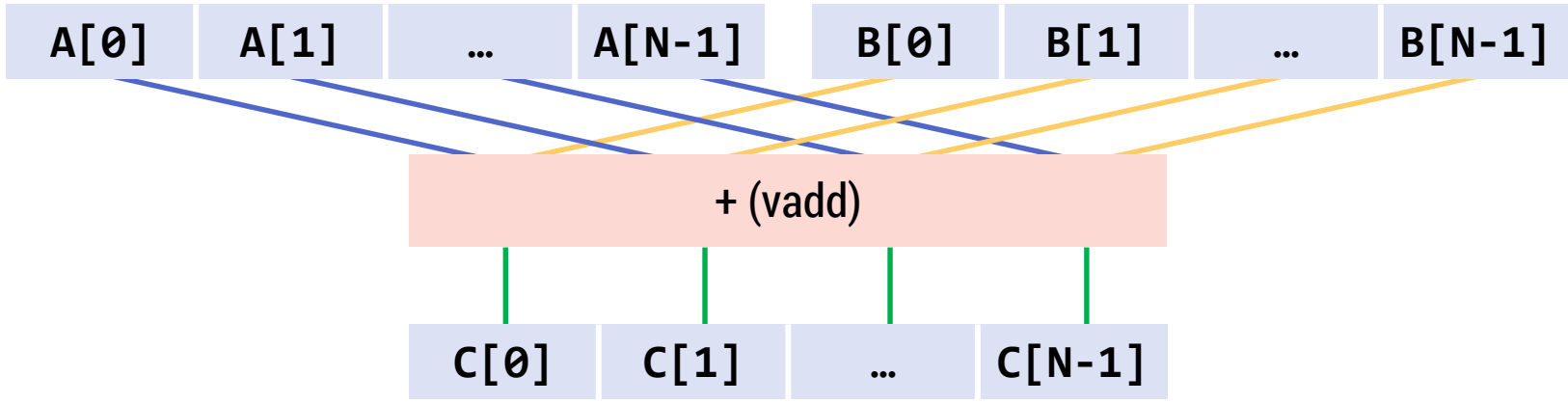
Скалярная операция

```
for (i = 0; i < N; i++)
    C[i] = A[i] + B[i]
```



Векторная операция

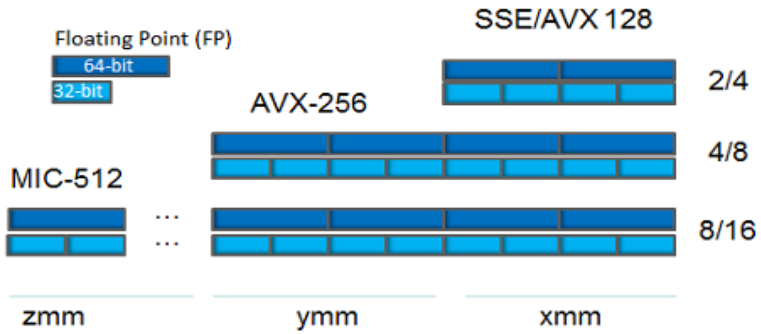
```
C[0..N-1] = A[0..N-1] + B[0..N-1]
```



SIMD-команды процессоров

- **Наборы векторных команд**
  - Intel MMX/SSE/AVX/AVX-3, AVX-512
  - IBM AltiVec
  - ARM NEON SIMD
  - MIPS MSA

Векторные регистры (Intel 64, Intel Xeon Phi)



- **Достигаемые ускорения**

Тип данных	**Intel SSE** (регистры 128 бит)	**Intel AVX** (регистры 256 бит)	**Intel AVX-512** (регистры 512 бит)	**ARMv8 Scalable Vector Extension** (регистры 128-2048 бит, RIKEN Post-K supercomputer, 2020)
`double`	x2	x4	x8	**x32**
`float`	x4	x8	x16	**x64**
`int`	x4	x8	x16	x64
`short int`	x8	x16	x32	x128

- **Причины снижения ускорения**
  - Адреса массивов не выравнены на заданную границу (32 байта для AVX и 64 байта для AVX-512)
  - Смешанное использование SSE- и AVX-команд (AVX-SSE Transition Penalties) [1]

[1] Konsor P. ***Avoiding AVX-SSE Transition Penalties*** // URL: https://software.intel.com/en-us/articles/avoiding-avx-sse-transition-penalties

Способы векторизации кода

Полный контроль, низкая переносимость

Простота использования, высокая переносимость

- **Ассемблерные вставки**

```
void add_sse(float *a, float *b, float *c) {
  __asm__ __volatile__ (
    "movaps (%[a]), %%xmm0 \n\t"
    "movaps (%[b]), %%xmm1 \n\t"
    "addps %%xmm1, %%xmm0 \n\t"
    "movaps %%xmm0, %[c] \n\t"
}
```

- **Интринсики (intrinsics)** –
  встроенные функции и типы данных компилятора

```
void add_sse(float *a, float *b, float *c) {
    __m128 t0, t1;
    t0 = _mm_load_ps(a);
    t1 = _mm_load_ps(b);
    t0 = _mm_add_ps(t0, t1);
    _mm_store_ps(c, t0);
}
```

- **SIMD-директивы** компиляторов,
  стандартов OpenMP, OpenACC

```
void f(double *a, double *b, double *c, int n) {
    #pragma omp simd
    for (int i = 0; i < n; i++)
        c[i] += a[i] * b[i];
}
```

- **Языковые расширения** (Intel Array Notation, Intel ISPC, Apple Swift SIMD)
  и библиотеки (C++17 SIMD, Boost.SIMD, SIMD.js)
- **Автоматическая векторизация компилятором**

```
$ gcc -ftree-vectorize ./vec.c
vec.c:13:5: note: loop vectorized
vec.c:18:5: note: not vectorized, possible dependence between data-refs
```

Набор тестовых циклов

1991

- **TSVC – Test Suite for Vectorizing Compilers [1]**
  (122 цикла на Fortran)

  Векторные ВС:
  Cray, NEC, IBM, DEC, Fujitsu, Hitachi

```
   do 1 nl = 1, 2*ntimes
      do 10 i = 2, n, 2
         a(i) = a(i-1) + b(i)
10    continue
1   continue
```

2011

- **ETSVC – Extended Test Suite for Vectorizing Compilers [2, 3]**
  (151 цикл на C)

  Наборы векторных инструкций:
  Intel SSE/AVX, IBM AltiVec, ARM NEON SIMD, MIPS MSA

```
for (int nl = 0; nl < 2*ntimes; nl++) {
    for (int i = 1; i < n; i += 2) {
        a[i] = a[i - 1] + b[i];
    }
}
```


[1] Levine D., Callahan D., Dongarra J. ***A Comparative Study of Automatic Vectorizing Compilers*** // Journal of Parallel Computing. 1991. Vol. 17. pp. 1223–1244.

[2] Maleki S., Gao Ya. Garzarán M.J., Wong T., Padua D.A. ***An Evaluation of Vectorizing Compilers*** // Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques (PACT-11), 2011. pp. 372–382.

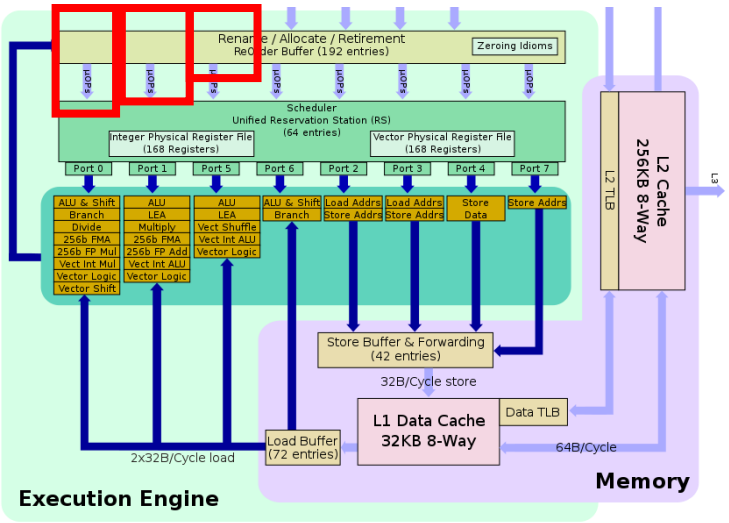
[3] ***Extended Test Suite for Vectorizing Compilers***. URL: http://polaris.cs.uiuc.edu/~maleki1/TSVC.tar.gz

Категории циклов ETSVC

Категория	Число циклов
Анализ зависимостей по данным (dependence analysis)	36
Анализ потока управления и трансформация циклов (vectorization)	52
Распознавание идиоматических конструкций (idiom recognition)	27
Полнота понимания языка программирования (language completeness)	23
Контрольные циклы (control loops)	13

Целевая архитектура

- **Двухпроцессорный NUMA-сервер**
  - **2 процессора Intel Xeon E5-2620 v4:**
    архитектура Intel 64, микроархитектура Broadwell, 8 ядер,
    Hyper-Threading включен, группа векторных АЛУ с поддержкой AVX 2.0
  - **Память**: 64 GiB, DDR4,
  - **Операционная система**: GNU/Linux CentOS 7.3 x86-64
    (ядро linux 3.10.0-514.2.2.el7)



https://en.wikichip.org/wiki/intel/microarchitectures/broadwell

Целевые компиляторы

Компилятор	Опции компиляции	Отключение векторизатора
GCC C/C++ 6.3.0	`-O3 -ffast-math -fivopts -march=native -fopt-info-vec -fopt-info-vec-missed -fno-tree-vectorize`	`-fno-tree-vectorize`
LLVM/Clang 3.9.1	`-O3 -ffast-math -fvectorize -Rpass=loop-vectorize -Rpass-missed=loop-vectorize -Rpass-analysis=loop-vectorize`	`-fno-vectorize`

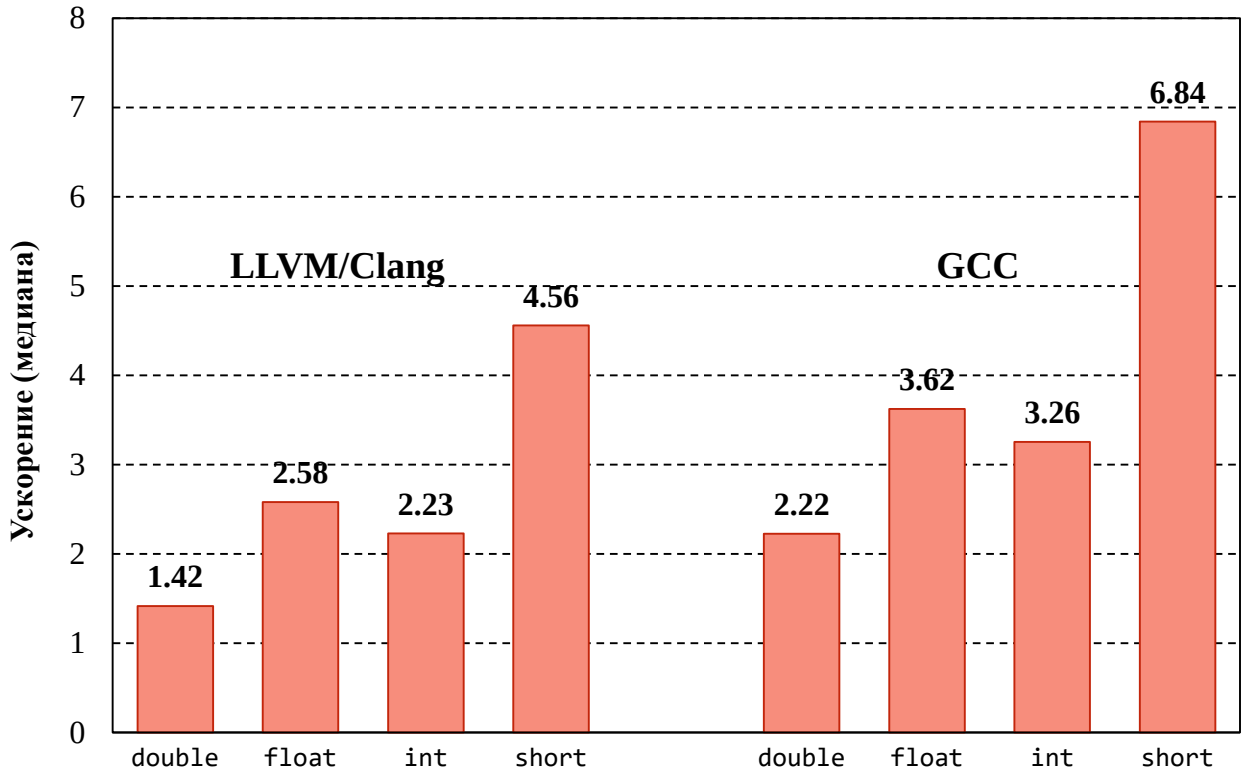
Результаты экспериментов

Классы невекторизованных циклов

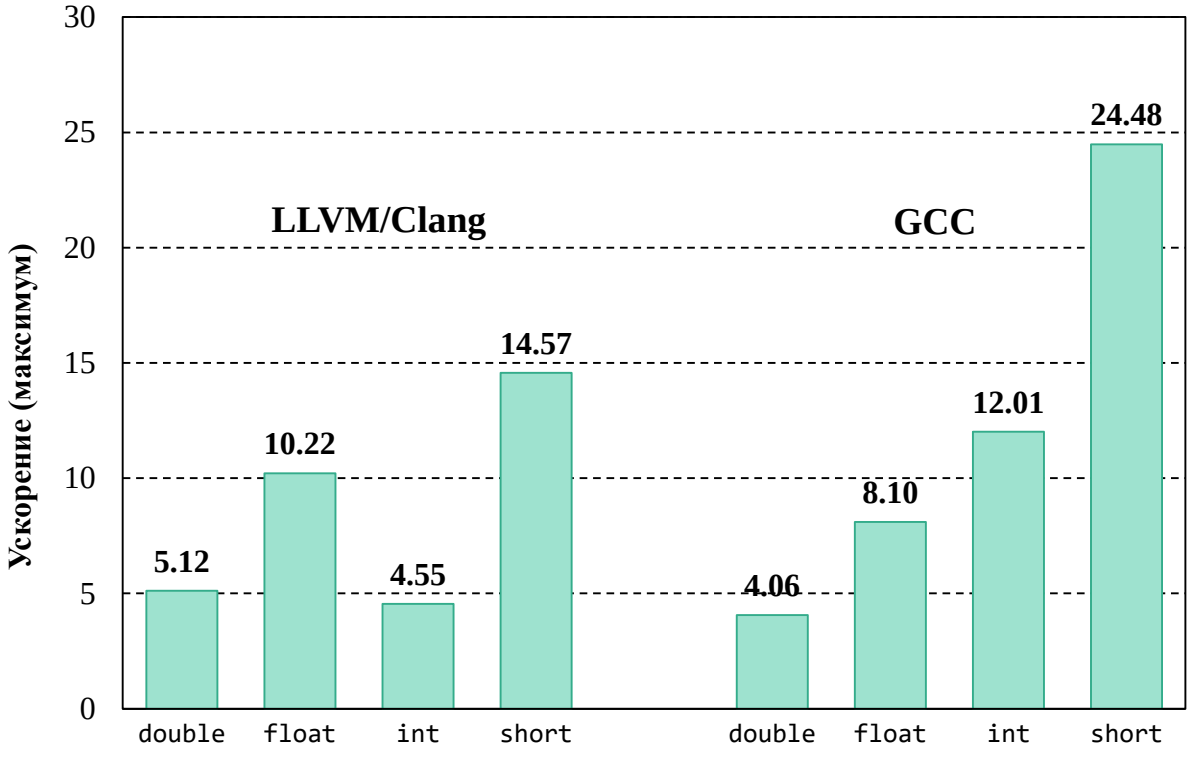
Категория / Подкатегория	**Общее число циклов**	**Число невекторизованных циклов**
Анализ зависимостей по данным (dependence analysis)	**36**	**14**
Анализ потока управления и трансформация циклов (vectorization)	**52**	**29**
Распознавание идиоматических конструкций (idiom recognition)	**27**	**15**
Полнота понимания языка программирования (language completeness)	**23**	**6**
Контрольные циклы (control loops)	**13**	**2**

66 циклов (39,7 %) не векторизованы ни одним из компиляторов!

Ускорение выполнения векторизованных циклов на процессоре Intel Xeon E5-2620 v4



Медиана ускорения



Максимальное ускорение

DGEMM(A[M][N], B[M][K], C[K][N])
A = A + B * C

$\langle i,j,k \rangle$	`for i = 0, M - 1, 1`
	`  for j = 0, N - 1, 1`
	`  for k = 0, K - 1, 1`
	`  A[i][j] += B[i][k] * C[k][j]`
		$\langle i,k,j \rangle$	`for i = 0, M - 1, 1`
			`  for k = 0, K - 1, 1`
			`  for j = 0, N - 1, 1`
			`  A[i][j] += B[i][k] * C[k][j]`
				Скалярные версии

DGEMM(A[M][N], B[M][K], C[K][N])
A = A + B * C

$\langle i, j, k \rangle$

```
for i = 0, M - 1, 1
  for j = 0, N - 1, 1
    for k = 0, K - 1, 1
      A[i][j] += B[i][k] * C[k][j]
```

$\langle i, k, j \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, 1
    for j = 0, N - 1, 1
      A[i][j] += B[i][k] * C[k][j]
```

Скалярные версии

$\langle i, j, \vec{k} \rangle$

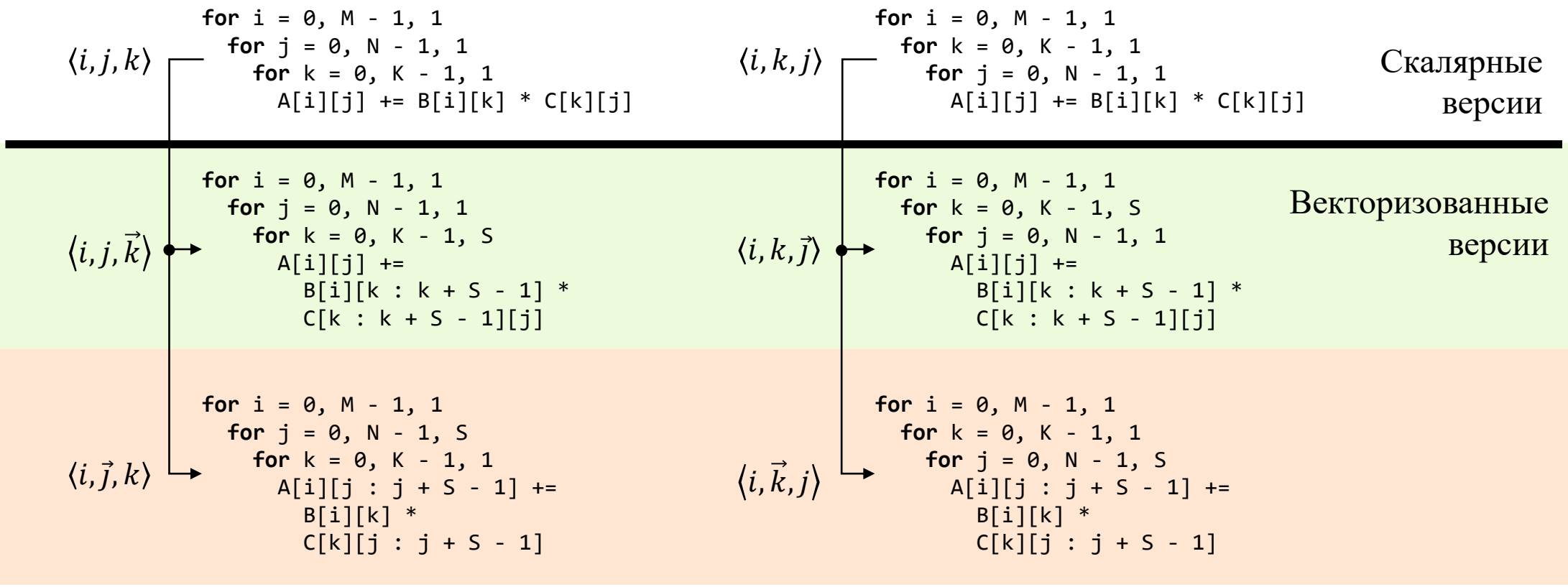
```
for i = 0, M - 1, 1
  for j = 0, N - 1, 1
    for k = 0, K - 1, S
      A[i][j] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j]
```

$\langle i, k, \vec{j} \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, S
    for j = 0, N - 1, 1
      A[i][j] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j]
```

Векторизованные версии

DGEMM(A[M][N], B[M][K], C[K][N])
A = A + B * C



DGEMM(A[M][N], B[M][K], C[K][N])
A = A + B * C

$\langle i, j, k \rangle$

```
for i = 0, M - 1, 1
  for j = 0, N - 1, 1
    for k = 0, K - 1, 1
      A[i][j] += B[i][k] * C[k][j]
```

$\langle i, k, j \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, 1
    for j = 0, N - 1, 1
      A[i][j] += B[i][k] * C[k][j]
```

Скалярные версии

Векторизованные версии

$\langle i, j, \vec{k} \rangle$

```
for i = 0, M - 1, 1
  for j = 0, N - 1, 1
    for k = 0, K - 1, S
      A[i][j] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j]
```

$\langle i, k, \vec{j} \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, S
    for j = 0, N - 1, 1
      A[i][j] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j]
```

$\langle i, \vec{j}, k \rangle$

```
for i = 0, M - 1, 1
  for j = 0, N - 1, S
    for k = 0, K - 1, 1
      A[i][j : j + S - 1] +=
        B[i][k] *
        C[k][j : j + S - 1]
```

$\langle i, \vec{k}, j \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, 1
    for j = 0, N - 1, S
      A[i][j : j + S - 1] +=
        B[i][k] *
        C[k][j : j + S - 1]
```

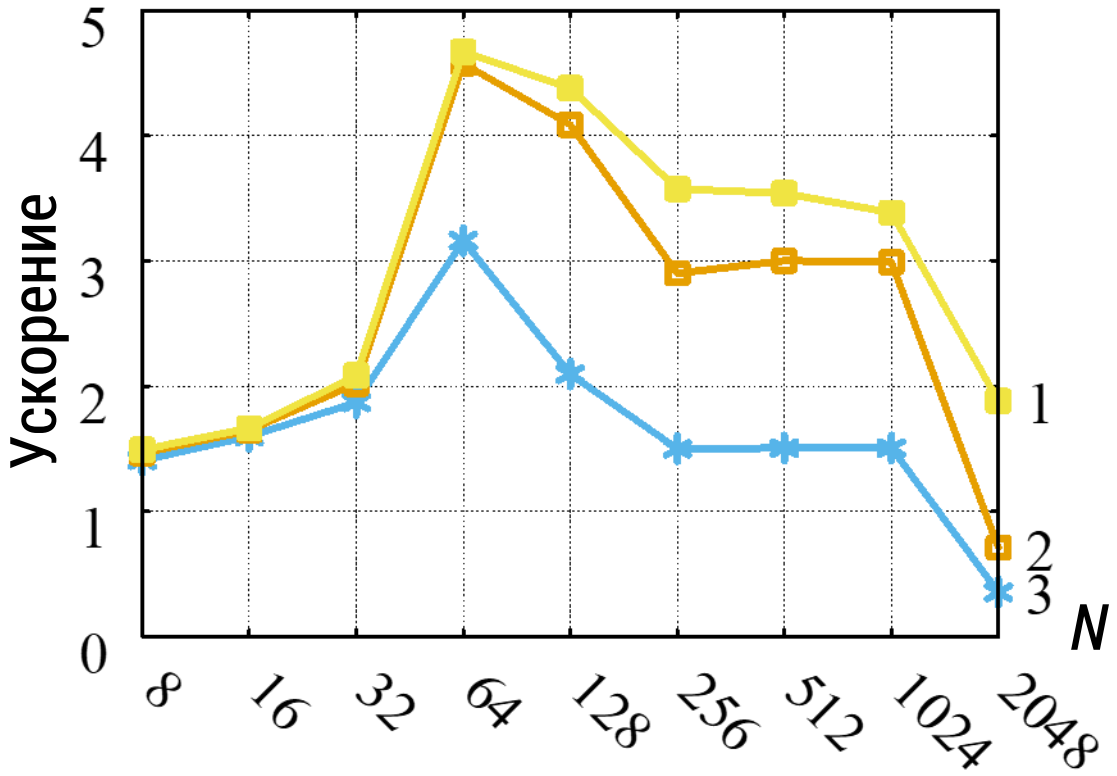
$\langle i, \vec{j}, \vec{k} \rangle$

```
for i = 0, M - 1, 1
  for j = 0, N - 1, S
    for k = 0, K - 1, S
      A[i][j : j + S - 1] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j : j + S - 1]
```

$\langle i, \vec{k}, \vec{j} \rangle$

```
for i = 0, M - 1, 1
  for k = 0, K - 1, S
    for j = 0, N - 1, S
      A[i][j : j + S - 1] +=
        B[i][k : k + S - 1] *
        C[k : k + S - 1][j : j + S - 1]
```

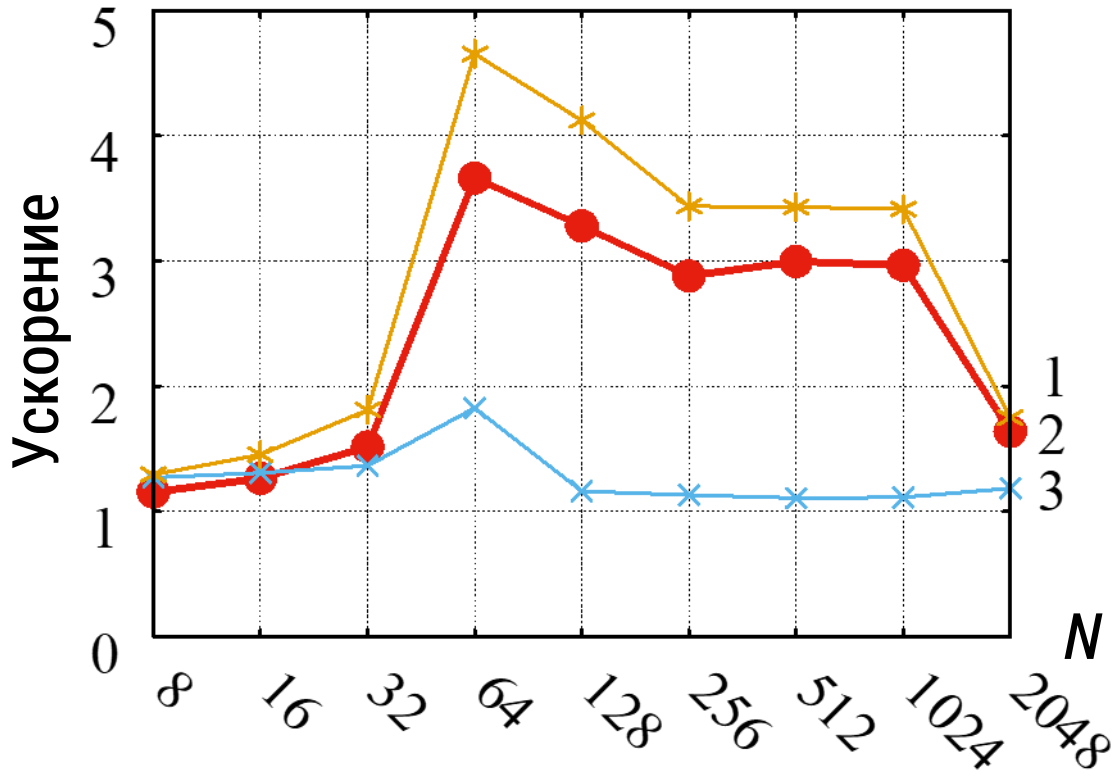
Ускорение тестов на микроархитектуре Haswell



1 – $< i, \vec{j}, \vec{k} >$ с непоследовательным доступом к элементам матрицы *C*

2 – $< i, \vec{j}, \vec{k} >$ с использованием команды *vbroadcastsd*

3 – $< i, \vec{j}, k >$ векторизация только среднего цикла



1 – $< i, \vec{k}, \vec{j} >$ с использованием команды *vbroadcastsd*

2 – $< i, k, \vec{j} >$ векторизация только внутреннего цикла

3 – $< i, \vec{k}, \vec{j} >$ с непоследовательным доступом к элементам матрицы *C*

Значения счетчиков производительности на микроархитектуре Haswell

Версия DGEMM	Ускоре-ние	L1d -load-misses	LLC-load-misses	mem-loads	mem-stores	cycles	instructions	CPI	branch-instructions	branch-misses
<i, j, k>:1	4.67	8684	0	86038	1034	93663	240100	0.39	17501	72
<i, j, k>:2	4.57	9884	0	100368	1034	97662	203236	0.48	8797	71
<i, j, k>:3	3.16	19336	0	198678	2060	213945	600612	0.36	66653	1095
<i, k, j>:1	4.65	8730	0	86038	16394	91136	222693	0.41	17501	72
<i, k, j>:2	3.66	9643	0	135190	65546	144023	549349	0.26	69725	72
<i, k, j>:3	1.83	9282	0	328728	65547	483118	1169893	0.41	83037	78

$< i, \vec{j}, \vec{k} >$: 1 – Векторизация с непоследовательным доступом к элементам матрицы *C*

$< i, \vec{j}, \vec{k} >$: 2 – Векторизация с использованием команды *vbroadcastsd*

$< i, \vec{j}, k >$: 3 – Векторизация только среднего цикла

$< i, \vec{k}, \vec{j} >$: 1 – Векторизация с использованием команды *vbroadcastsd*

$< i, k, \vec{j} >$: 2 – Векторизация только внутреннего цикла

$< i, \vec{k}, \vec{j} >$: 3 – Векторизация с непоследовательным доступом к элементам матрицы *C*

Значения счетчиков производительности на микроархитектуре Haswell

Версия DGEMM	Ускоре-ние	L1d -load-misses	LLC-load-misses	mem-loads	mem-stores	cycles	instructions	CPI	branch-instructions	branch-misses
<i, j, k>:1	4.67	8684	0	86038	1034	93663	240100	0.39	17501	72
<i, j, k>:2	4.57	9884	0	100368	1034	97662	203236	0.48	8797	71
<i, j, k>:3	3.16	19336	0	198678	2060	213945	600612	0.36	66653	1095
<i, k, j>:1	4.65	8730	0	86038	16394	91136	222693	0.41	17501	72
<i, k, j>:2	3.66	9643	0	135190	65546	144023	549349	0.26	69725	72
<i, k, j>:3	1.83	9282	0	328728	65547	483118	1169893	0.41	83037	78

$< i, \vec{j}, \vec{k} >$: 1 – Векторизация с непоследовательным доступом к элементам матрицы *C*

$< i, \vec{j}, \vec{k} >$: 2 – Векторизация с использованием команды *vbroadcastsd*

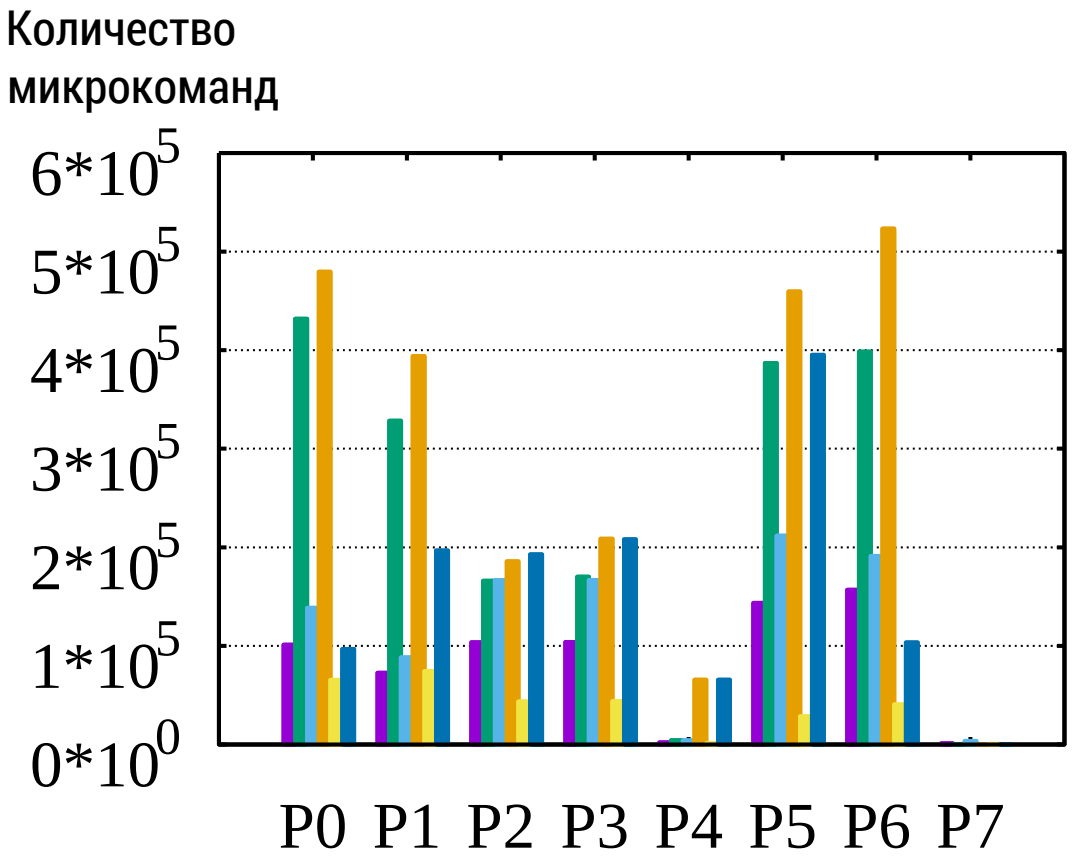
$< i, \vec{j}, k >$: 3 – Векторизация только среднего цикла

$< i, \vec{k}, \vec{j} >$: 1 – Векторизация с использованием команды *vbroadcastsd*

$< i, k, \vec{j} >$: 2 – Векторизация только внутреннего цикла

$< i, \vec{k}, \vec{j} >$: 3 – Векторизация с непоследовательным доступом к элементам матрицы *C*

Загруженность функциональных устройств на микроархитектуре Haswell

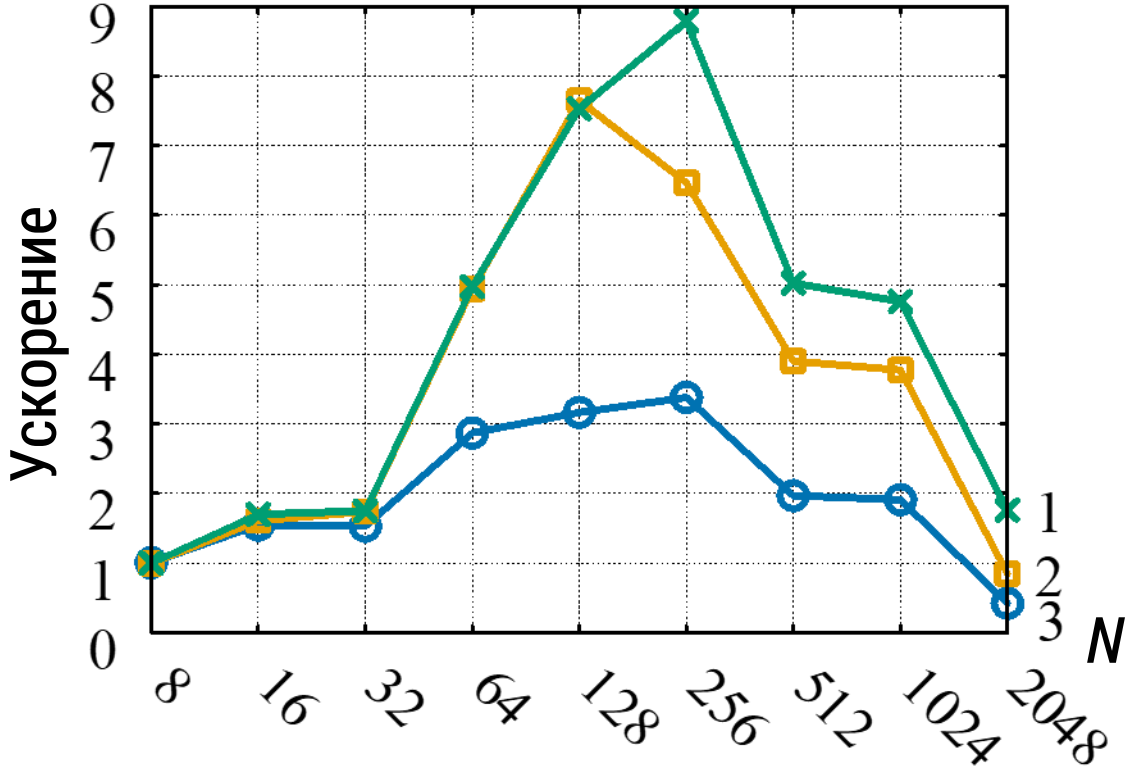


Размер матриц: 64×64 элемента типа **double**

- – Векторизация с непоследовательным доступом к элементам матрицы *C*
- – Векторизация с использованием команды *vbroadcastsd*
- – Векторизация только среднего цикла
- – Векторизация с использованием команды *vbroadcastsd*
- – Векторизация только внутреннего цикла
- – Векторизация с непоследовательным доступом к элементам матрицы *C*

P[0-7] – Функциональные устройства, выполняющие микрокоманды (АЛУ)

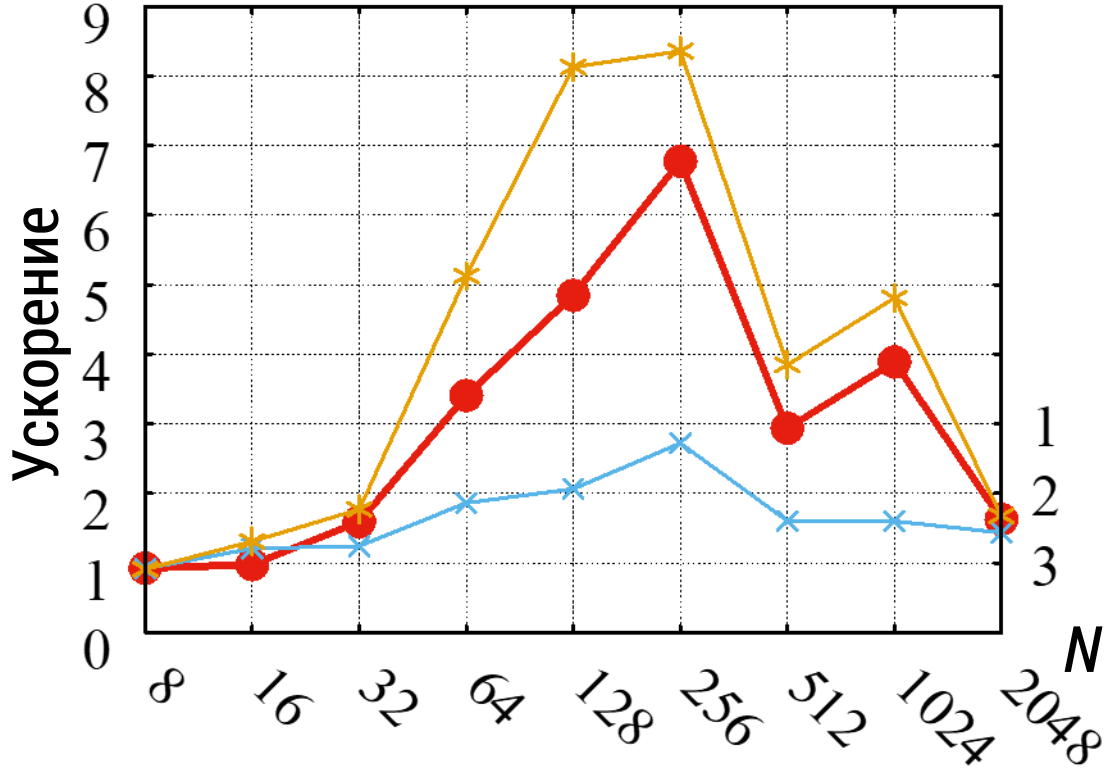
Ускорение тестов на микроархитектуре Broadwell



1 – $< i, \vec{j}, \vec{k} >$ с непоследовательным доступом к элементам матрицы *C*

2 – $< i, \vec{j}, \vec{k} >$ с использованием команды *vbroadcastsd*

3 – $< i, \vec{j}, k >$ векторизация только среднего цикла



1 – $< i, \vec{k}, \vec{j} >$ с использованием команды *vbroadcastsd*

2 – $< i, k, \vec{j} >$ векторизация только внутреннего цикла

3 – $< i, \vec{k}, \vec{j} >$ с непоследовательным доступом к элементам матрицы *C*

Значения счетчиков производительности на микроархитектуре Broadwell

Версия DGEMM	Ускорение	L1d – load-misses	LLC-load-misses	mem-loads	mem-stores	cycles	instructions	CPI	branch-instructions	branch-misses
<i, j, k>:1	8.79	2115816	0	5308466	16396	9850108	14059112	0.7	1065249	267
<i, j, k>:2	6.46	2143096	0	6324281	16396	13456733	12682856	1.06	532769	275
<i, j, k>:3	3.38	4263691	3456	12615758	32780	25802907	37914476	0.68	4210981	16901
<i, k, j>:1	8.36	2118641	0	5308466	1048588	10355924	13780585	0.75	1065249	270
<i, k, j>:2	6.77	2098595	1826	8454194	4194316	12790121	33949289	0.38	4260129	520
<i, k, j>:3	2.72	2178667	185	20987997	4194317	32114941	74548847	0.43	5259559	16657

$< i, \vec{j}, \vec{k} >$: 1 – Векторизация с непоследовательным доступом к элементам матрицы *C*

$< i, \vec{j}, \vec{k} >$: 2 – Векторизация с использованием команды *vbroadcastsd*

$< i, \vec{j}, k >$: 3 – Векторизация только среднего цикла

$< i, \vec{k}, \vec{j} >$: 1 – Векторизация с использованием команды *vbroadcastsd*

$< i, k, \vec{j} >$: 2 – Векторизация только внутреннего цикла

$< i, \vec{k}, \vec{j} >$: 3 – Векторизация с непоследовательным доступом к элементам матрицы *C*

Значения счетчиков производительности на микроархитектуре Broadwell

Версия DGEMM	Ускорение	L1d – load-misses	LLC-load-misses	mem-loads	mem-stores	cycles	instructions	CPI	branch-instructions	branch-misses
<i, j, k>:1	8.79	2115816	0	5308466	16396	9850108	14059112	0.7	1065249	267
<i, j, k>:2	6.46	2143096	0	6324281	16396	13456733	12682856	1.06	532769	275
<i, j, k>:3	3.38	4263691	3456	12615758	32780	25802907	37914476	0.68	4210981	16901
<i, k, j>:1	8.36	2118641	0	5308466	1048588	10355924	13780585	0.75	1065249	270
<i, k, j>:2	6.77	2098595	1826	8454194	4194316	12790121	33949289	0.38	4260129	520
<i, k, j>:3	2.72	2178667	185	20987997	4194317	32114941	74548847	0.43	5259559	16657

$< i, \vec{j}, \vec{k} >$: 1 – Векторизация с непоследовательным доступом к элементам матрицы *C*

$< i, \vec{j}, \vec{k} >$: 2 – Векторизация с использованием команды *vbroadcastsd*

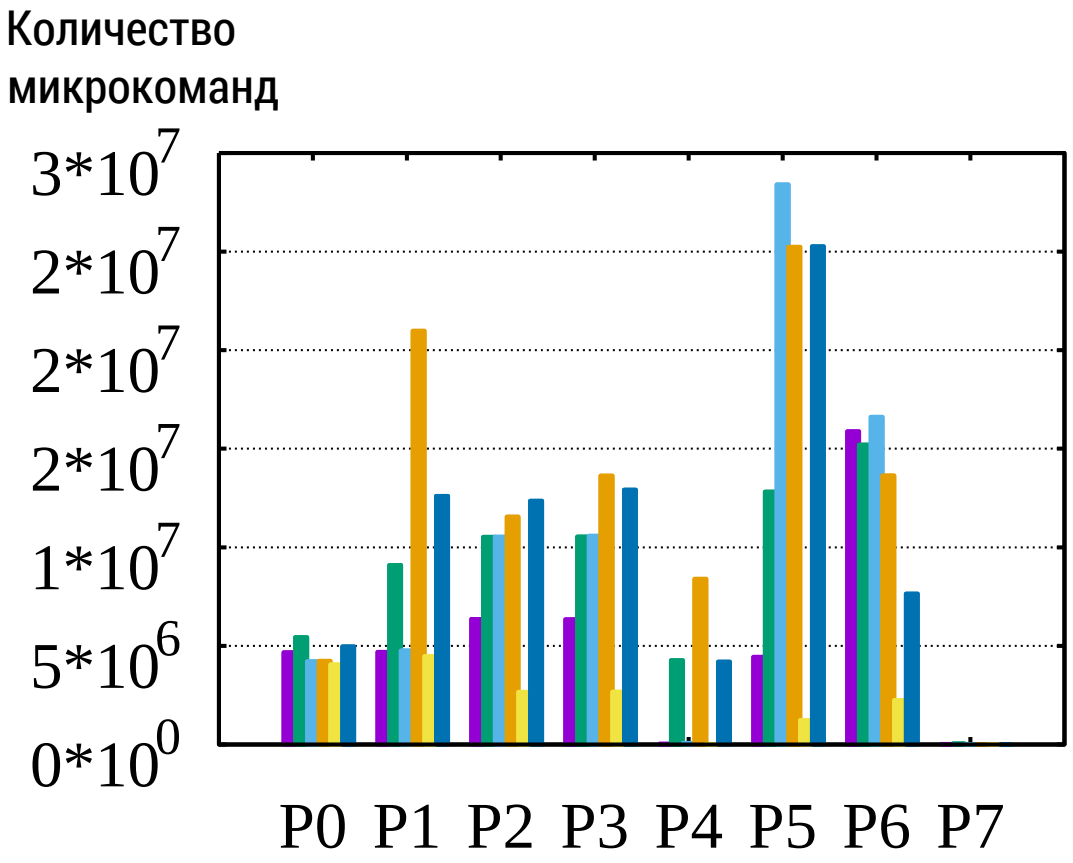
$< i, \vec{j}, k >$: 3 – Векторизация только среднего цикла

$< i, \vec{k}, \vec{j} >$: 1 – Векторизация с использованием команды *vbroadcastsd*

$< i, k, \vec{j} >$: 2 – Векторизация только внутреннего цикла

$< i, \vec{k}, \vec{j} >$: 3 – Векторизация с непоследовательным доступом к элементам матрицы *C*

Загруженность функциональных устройств на микроархитектуре Broadwell



Размер матриц: 256×256 элемента типа **double**

- – Векторизация с непоследовательным доступом к элементам матрицы *C*
- – Векторизация с использованием команды *vbroadcastsd*
- – Векторизация только среднего цикла
- – Векторизация с использованием команды *vbroadcastsd*
- – Векторизация только внутреннего цикла
- – Векторизация с непоследовательным доступом к элементам матрицы *C*

P[0-7] – Функциональные устройства, выполняющие микрокоманды (АЛУ)

Спасибо за внимание!

Ольга Владимировна Молдованова [1,2] **Иван Иванович Кулагин** [1,2] **Михаил Георгиевич Курносов** [1,2]

ovm@sibguti.ru, ovm@isp.nsc.ru ivan.i.kulagin@gmail.com WWW: www.mkurnosov.net

[1] Кафедра вычислительных систем, Сибирский государственный университет телекоммуникаций и информатики, Новосибирск

[2] Лаборатория вычислительных систем, Институт физики полупроводников им. А.В. Ржанова СО РАН, Новосибирск

Тринадцатая международная азиатская школа-семинар «Проблемы оптимизации сложных систем» (ПОСС-2017)

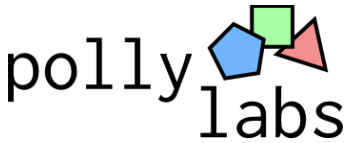
в рамках международной мультиконференции IEEE SIBIRCON 2017

18-22 сентября 2017 г., Академгородок, Новосибирск, Россия

Направление дальнейшей работы

- Анализ известных методов векторизации и распараллеливания циклов (полиэдральные модели: GCC Graphite, LLVM/Clang PollyLabs)

GRAPHITE



- Разработка методов векторизации установленного класса проблемных циклов из пакета ETSVC

- Анализ возможностей применения JIT-компиляции и оптимизации по результатам профилирования (profile-guided optimization) для автоматической векторизации кода

Категория «Анализ зависимостей по данным» (dependence analysis)

Подкатегория «Линейная зависимость по данным» (linear dependence)

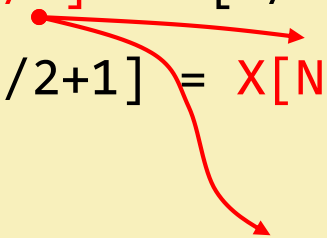
Невекторизованный цикл s1113

```
for (int i = 0; i < N; i++)
    X[i] = X[N/2] + Y[i];
```

- Зависимость по данным вида «чтение после записи» (read-after-write, RAW), начиная с итерации *N* / 2 + 1

Раскрутка цикла s1113 по итерациям

```
X[0] = X[N/2] + Y[0];
X[1] = X[N/2] + Y[1];
…
X[N/2] = X[N/2] + Y[N/2];
X[N/2+1] = X[N/2] + Y[N/2+1];
…
X[N-1] = X[N/2] + Y[N-1];
```



Категория «Анализ зависимостей по данным» (dependence analysis)

Подкатегория «Линейная зависимость по данным» (linear dependence)

Невекторизованный цикл s1113	Возможная трансформация цикла s1113

`for (int i = 0; i < N; i++)`
`  X[i] = X[N/2] + Y[i];`
	`int k = N / 2;`
	
	`for (int i = 0; i <= k; i++)`
	`  X[i] = X[k] + Y[i];`
	
	`for (int i = k + 1; i < N; i++)`
	`  X[i] = X[k] + Y[i];`

- Зависимость по данным вида «чтение после записи» (read-after-write, RAW), начиная с итерации *N* / 2 + 1

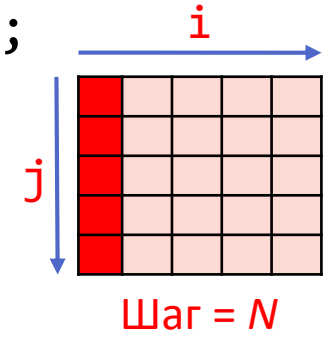
- ***Распределение путем разбиения имен*** (fission by name)
- Ускорение в **2** раза для типа `double` и компилятора ICC на Intel Xeon E5-2620 v4

Категория «Анализ зависимостей по данным» (dependence analysis)

Подкатегория «Распознавание индуктивной переменной» (induction variable recognition)

Невекторизованный цикл s126	Возможная трансформация цикла s126

```
int k = 1;
for (int i = 0; i < N; i++) {
    for (int j = 1; j < N; j++) {
        X[j][i] = X[j - 1][i] +
                  Y[k - 1] *
                  Z[j][i];
        ++k;
    }
    ++k;
}
```




```
for (int j = 1; j < N; j++) {
    for (int i = 0; i < N; i++) {
        X[j][i] = X[j - 1][i] +
                  Y[i * 4 + j - 1] *
                  Z[j][i];
    }
}
```

- Индуктивная переменная *k*
- Внешний цикл осуществляет проход по столбцам матриц *X* и *Z*, а внутренний – по строкам

- ***Перестановка циклов и удаление индуктивной переменной k***
- Ускорение в **8.5** раз для типа `double` и компилятора ICC на Intel Xeon E5-2620 v4

Категория «Анализ зависимостей по данным» (dependence analysis)

Подкатегория «Условные и безусловные переходы» (control flow)

Невекторизованный цикл s161

```
for (int i = 0; i < N - 1; ++i)
{
     if (Y[i] < 0)
        goto L20;

     X[i] = Z[i] + V[i] * W[i];
     goto L10;

L20: Z[i + 1] = X[i] + V[i] * V[i];
L10: ;
}
```

Раскрутка цикла s161 по итерациям

```
i = 0:
        if (Y[0] < 0)
S1^0:      Z[1] = X[0] + V[0] * V[0];
        else
S2^0:      X[0] = Z[0] + V[0] * W[0];

i = 1:                                  S1 δ< S2
        if (Y[1] < 0)
S1^1:      Z[2] = X[1] + V[1] * V[1];
        else
S2^1:      X[1] = Z[1] + V[1] * W[1];
```

Возможная зависимость выражения S_2 на итерации i от выражения S_1 на итерации i - 1
вида «чтение после записи» (read-after-write, RAW)

Категория «Анализ зависимостей по данным» (dependence analysis)

Подкатегория «Переменные в границах цикла или шаге выполнения итераций» (symbolics)

Невекторизованный цикл s172

```
void s172(int n1, int n3)
{
    for (int i = n1 - 1; i < N; i += n3)
        X[i] += Y[i];
}
```

Переменные, используемые в качестве нижней и(или) верхней границы цикла и(или) шага выполнения итераций

Категория «Анализ потока управления и трансформация циклов» (vectorization)

Подкатегория «Растягивание скаляров и массивов» (scalar and array expansion)

Невекторизованный цикл s257	Возможная трансформация цикла s257

```
    for (int i = 1; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
S1:         X[i] = Y[j][i] - X[i - 1];
S2:         Y[j][i] = X[i] + Z[j][i];
        }
    }
```

$S_1\ \delta_{<}\ S_1 \quad S_1\ \delta_{=}\ S_2 \quad S_1\ \bar{\delta}_{=,=}\ S_2$

```
for (int i = 1; i < N; i++)
    for (int j = 0; j < N; j++)
        X[i] = Y[j][i] - X[i - 1];

for (int i = 1; i < N; i++)
    for (int j = 0; j < N; j++)
        Y[j][i] = Y[j][i] - X[i - 1] +
                  Z[j][i];
```

- Зависимость выражения S_1 на итерации *i* от S_1 на итерации *i* - 1 вида «чтение после записи» (read-after-write)
- Зависимость выражения S_2 на итерации *i* от S_1 на той же итерации вида «чтение после записи» (read-after-write)
- Зависимость выражения S_2 на итерациях *i, j* от S_1 на тех же итерациях вида «запись после чтения» (write-after-read)

- ***Расщепление тела цикла*** (loop fission, loop distribution)
- Ускорение в **9.3** раза для типа `double` и компилятора ICC на Intel Xeon E5-2620 v4

Категория «Распознавание идиоматических конструкций» (idiom recognition)

Подкатегория «Рекуррентности» (recurrences)

Невекторизованный цикл s322

```
     for (int i = 2; i < N; i++)
     {
S1:      X[i] = X[i] + X[i-1] *
                Y[i] + X[i-2] * Z[i];
     }
```

$S_1\ \delta_{<}\ S_1$

Зависимость выражения S_1 на итерации i от выражения S_1 на итерациях $i - 1$ и $i - 2$
вида «чтение после записи» (read-after-write, RAW)

Категория «Распознавание идиоматических конструкций» (idiom recognition)

Подкатегория «Свертка цикла» (loop rerolling)

Невекторизованный цикл s353

```
void s353(int* __restrict__ ip) {
    double alpha = Z[0];
    for (int i = 0; i < N; i += 5) {
        X[i]   += alpha * Y[ip[i]];
        X[i+1] += alpha * Y[ip[i+1]];
        X[i+2] += alpha * Y[ip[i+2]];
        X[i+3] += alpha * Y[ip[i+3]];
        X[i+4] += alpha * Y[ip[i+4]];
    }
}
```

Косвенная адресация при обращении к элементам массива *Y*[*ip*[*i*]]

Категория «Распознавание идиоматических конструкций» (idiom recognition)

Подкатегория «Редукции» (reductions)

Невекторизованный цикл s31111

```
double test(double* A) {
    double s = (double)0.;
    for (int i = 0; i < 4; i++)
        s += A[i];
    return s;
}

void s31111() {
    double sum;
    for (int i = 0; i < N; i++) {
        sum = (double)0.;
        sum += test(X);
        sum += test(&X[4]);
        sum += test(&X[8]);
        sum += test(&X[12]);
        sum += test(&X[16]);
        sum += test(&X[20]);
        sum += test(&X[24]);
        sum += test(&X[28]);
    }
}
```

Наличие вызова функции в теле цикла

Категория «Полнота понимания языка программирования» (language completeness)

Подкатегория «Прерывание вычислений в цикле» (nonlocal GOTO)

Невекторизованный цикл s481

```
for (int i = 0; i < N; i++)
{
    if (V[i] < (double)0.)
       exit(0);

    X[i] += Y[i] * Z[i];
}
```

Наличие вызова функции `exit` в теле цикла

Результаты автоматической векторизации циклов

(Intel 64, тип данных double)

V	Цикл векторизован полностью	**M**	Мультиверсионность	**NI**	Невозможно вычислить количество итераций	**OL**	Значение не может быть использовано за пределами цикла
PV	Цикл векторизован частично	**BO**	Неподходящая операция	**CF**	Невозможно определить направление потока управления	**UV**	Векторизатор не может понять поток управления в цикле
RV	Остаток цикла не векторизован	**AP**	Сложный шаблон доступа к элементам массива	**SS**	Цикл не подходит для векторной записи по несмежным адресам	**SW**	Наличие оператора `switch` в цикле
IF	Векторизация возможна, но не эффективна	**R**	Значение, которое не может быть идентифицировано как результат редукции, используется вне цикла	**ME**	Цикл с несколькими выходами невозможно векторизовать	**US**	Неподдерживаемое использование в выражении
D	Зависимость по данным препятствует векторизации	**IL**	Переменная-счетчик внутреннего цикла не является инвариантом	**FC**	Цикл содержит вызовы функций или данные, которые невозможно проанализировать	**GS**	В базовом блоке нет сгруппированных операций записи

Цикл S235 векторизован GCC и PGI

	S000	S111	S1111	S112	S1112	S113	S1113	S114	S115	S1115	S116	S118	S119	S1119	S121	S122	S123	S124	S125	S126	S127	S128	S131	S132	S141	S151	S152	S161	S1161	S162	S171	S172	S173	S174	S175	S176	S211	S212	S1213	S221	S1221	S222	S231	S232	S1232	S233	S2233	S235	S241	S242
ICC	V	IF	V	IF	V	V	D	V	V	V	V	D	V	D	V	D	D	V	V	D	V	V	V	V	D	V	D	D	V	M	D	D	V	V	D	V	V	V	V	PV	V	PV	M	D	IF	V	V	PV	PV	PV
PGI	V	IF	V	V	V	V	D	V	V	V	D	V	V	V	V	NI	D	D	V	D	V	D	V	V	D	FC	FC	D	V	D	D	NI	V	V	NI	V	D	D	D	D	D	D	V	D	IF	V	V	V	D	D
LLVM	V	IF	V	D	V	D	D	D	D	IF	R	R	V	V	V	NI	CF	CF	V	R	IF	IF	V	V	R	V	V	CF	CF	V	V	NI	V	V	NI	V	R	R	R	R	V	R	R	R	IF	R	R	R	D	R
GCC	V	V	V	V	V	V	V	BO	V	AP	D	IL	V	V	V	NI	SS	V	V	AP	V	V	V	V	IL	V	M	M	M	M	V	NI	V	V	NI	V	D	D	D	D	V	D	V	IL	IL	D	M	V	D	D

Цикл:

	S243	S244	S1244	S2244	S251	S1251	S2251	S3251	S252	S253	S254	S255	S256	S257	S258	S261	S271	S272	S273	S274	S275	S2275	S276	S277	S278	S279	S1279	S2710	S2711	S2712	S281	S1281	S291	S292	S293	S2101	S2102	S2111	S311	S31111	S312	S313	S314	S315	S316	S317	S318	S319	S3110	S13110
ICC	PV	PV	PV	V	V	V	D	V	V	V	V	D	D	D	D	V	V	V	V	V	V	PV	V	D	V	V	V	V	V	V	D	V	V	V	D	IF	PV	D	V	IF	V	V	V	IF	V	V	V	V	D	V
PGI	D	D	D	D	V	V	V	V	V	V	V	V	D	D	V	D	V	V	V	V	D	V	IF	D	V	V	V	V	V	V	D	V	V	V	D	IF	V	V	V	FC	V	V	V	V	V	V	V	V	V	V
LLVM	V	D	D	V	V	V	R	V	V	CF	V	R	R	OL	R	R	CF	CF	CF	CF	R	IF	UV	CF	CF	CF	CF	CF	CF	CF	D	V	R	R	D	IF	IF	R	V	FC	V	V	V	R	V	V	R	V	R	V
GCC	D	D	D	V	V	V	US	V	US	V	FC	US	AP	BO	US	D	V	V	V	V	CF	V	V	D	M	CF	CF	V	V	V	V	V	US	US	V	IF	IF	D	V	FC	V	V	V	CF	V	V	US	V	V	V

	S3111	S3112	S3113	S321	S322	S323	S331	S332	S341	S342	S343	S351	S1351	S352	S353	S421	S1421	S422	S423	S424	S431	S441	S442	S443	S451	S452	S453	S461	S481	S482	S491	S4112	S4113	S4114	S4115	S4116	S4117	S4121	va	vag	vas	vif	vpv	vtv	vpvtv	vpvts	vpvpv	vtvtv	vsumr	vdotr	vbor
ICC	V	D	V	D	D	D	V	ME	D	D	D	V	V	V	IF	V	V	M	M	M	V	V	V	V	V	V	V	V	ME	ME	IF	V	IF	V	V	V	V	V	V	V	IF	V	V	V	V	V	V	V	V	V
PGI	V	V	V	D	D	D	V	ME	V	V	V	V	V	V	IF	V	V	V	V	V	V	V	V	V	V	IF	V	V	FC	ME	V	V	IF	D	V	V	V	V	V	IF	V	V	V	V	V	V	V	V	V	V
LLVM	R	R	V	R	R	R	R	UV	CF	CF	CF	IF	V	IF	IF	V	V	V	V	V	V	CF	SW	CF	V	V	R	V	UV	NI	IF	IF	IF	IF	IF	IF	IF	V	FC	IF	IF	CF	V	V	V	V	V	V	V	V	V
GCC	V	US	V	D	D	D	US	CF	SS	US	SS	V	V	GS	BO	V	V	V	V	V	V	CF	CF	V	BO	V	V	V	CF	CF	SS	V	SS	V	V	V	V	V	CF	V	SS	V	V	V	V	V	V	V	V	V	V

Результаты автоматической векторизации циклов

(Intel Xeon Phi, тип данных double)

V	Цикл векторизован полностью	**M**	Мультиверсионность	**NI**	Невозможно вычислить количество итераций	**OL**	Значение не может быть использовано за пределами цикла
PV	Цикл векторизован частично	**BO**	Неподходящая операция	**CF**	Невозможно определить направление потока управления	**UV**	Векторизатор не может понять поток управления в цикле
RV	Остаток цикла не векторизован	**AP**	Сложный шаблон доступа к элементам массива	**SS**	Цикл не подходит для векторной записи по несмежным адресам	**SW**	Наличие оператора `switch` в цикле
IF	Векторизация возможна, но не эффективна	**R**	Значение, которое не может быть идентифицировано как результат редукции, используется вне цикла	**ME**	Цикл с несколькими выходами невозможно векторизовать	**US**	Неподдерживаемое использование в выражении
D	Зависимость по данным препятствует векторизации	**IL**	Переменная-счетчик внутреннего цикла не является инвариантом	**FC**	Цикл содержит вызовы функций или данные, которые невозможно проанализировать	**GS**	В базовом блоке нет сгруппированных операций записи

Intel C/C++ Compiler 17.0 (`-mmic`) native mode

	S000	S111	S1111	S112	S1112	S113	S1113	S114	S115	S1115	S116	S118	S119	S1119	S121	S122	S123	S124	S125	S126	S127	S128	S131	S132	S141	S151	S152	S161	S1161	S162	S171	S172	S173	S174	S175	S176	S211	S212	S1213	S221	S1221	S222	S231	S232	S1232	S233	S2233	S235	S241	S242
double	V	V	V	V	V	V	D	V	V	V	PV	D	V	D	V	D	D	V	V	D	V	V	V	V	D	V	D	D	V	M	D	D	V	V	D	V	V	V	V	PV	RV	PV	M	D	IF	V	V	PV	PV	D
float	V	V	V	V	V	V	D	V	V	V	PV	D	V	D	V	D	D	V	V	D	V	V	V	V	D	V	D	D	V	M	D	D	V	V	D	V	V	V	V	PV	PV	PV	M	D	V	V	V	PV	PV	PV

	S243	S244	S1244	S2244	S251	S1251	S2251	S3251	S252	S253	S254	S255	S256	S257	S258	S261	S271	S272	S273	S274	S275	S2275	S276	S277	S278	S279	S1279	S2710	S2711	S2712	S281	S1281	S291	S292	S293	S2101	S2102	S2111	S311	S31111	S312	S313	S314	S315	S316	S317	S318	S319	S3110	S13110
double	PV	PV	PV	V	V	V	D	V	V	V	V	D	D	D	D	V	V	V	V	V	V	PV	V	D	V	V	V	V	V	V	D	V	V	V	D	V	PV	D	V	IF	V	V	V	V	V	V	V	V	D	V
float	PV	PV	PV	V	V	V	D	V	V	V	V	D	D	D	D	V	V	V	V	V	V	PV	V	D	V	V	V	V	V	V	D	V	V	V	D	V	PV	D	V	IF	V	V	V	V	V	V	V	V	D	V

	S3111	S3112	S3113	S321	S322	S323	S331	S332	S341	S342	S343	S351	S1351	S352	S353	S421	S1421	S422	S423	S424	S431	S441	S442	S443	S451	S452	S453	S461	S481	S482	S491	S4112	S4113	S4114	S4115	S4116	S4117	S4121	va	vag	vas	vif	vpv	vtv	vpvtv	vpvts	vpvpv	vtvtv	vsumr	vdotr	vbor
double	V	D	V	D	D	D	V	ME	V	V	V	V	PV	V	V	V	V	M	M	M	V	V	V	V	V	V	V	V	ME	ME	V	V	V	PV	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
float	V	D	V	D	D	D	V	ME	V	V	V	V	V	V	V	V	V	M	M	M	V	V	V	V	V	V	V	V	ME	ME	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V

Сравнение результатов с предыдущими работами

2011 г. [1]		2017 г.	
Intel C/C++ 12.0	90 циклов (59.6 %)	**Intel C/C++ 17.0**	95 циклов (62.9 %)
GCC C/C++ 4.7.0	59 циклов (39 %)	**GCC C/C++ 6.3.0**	79 циклов (52.3 %)

[1] Maleki S., Gao Ya. Garzarán M.J., Wong T., Padua D.A. ***An Evaluation of Vectorizing Compilers*** // Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques (PACT-11), 2011. pp. 372–382.

Ссылки на литературу

1. Maleki S., Gao Ya. Garzarán M.J., Wong T., Padua D.A. An Evaluation of Vectorizing Compilers // Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques (PACT'11), 2011. pp. 372–382.
2. Extended Test Suite for Vectorizing Compilers. URL: http://polaris.cs.uiuc.edu/~maleki1/TSVC.tar.gz.
3. Callahan D., Dongarra J., Levine D. Vectorizing Compilers: A Test Suite and Results // Proc. of the ACM/IEEE conf. on Supercomputing (Supercomputing'88), 1988. pp. 98–105.
4. Levine D., Callahan D., Dongarra J. A Comparative Study of Automatic Vectorizing Compilers // Journal of Parallel Computing. 1991. Vol. 17. pp. 1223–1244.
5. Konsor P. Avoiding AVX-SSE Transition Penalties. URL: https://software.intel.com/en-us/articles/avoiding-avx-sse-transition-penalties.
6. Jibaja I., Jensen P., Hu N., Haghighat M., McCutchan J., Gohman D., Blackburn S., McKinley K. Vector Parallelism in JavaScript: Language and Compiler Support for SIMD // Proc. of the Int. Conf. on Parallel Architecture and Compilation (PACT-2015). 2015. pp. 407–418.
7. Векторизация программ: теория, методы, реализация. Сб. статей: Пер. с англ. и нем. М.: Мир, 1991. 275 с.
8. Metzger R.C., Wen Zh. Automatic Algorithm Recognition and Replacement: A New Approach to Program Optimization. MIT Press. 2000. 219 p.
9. Rohou E., Williams K., Yuste D. Vectorization Technology To Improve Interpreter Performance // ACM Transactions on Architecture and Code Optimization. 2013. 9 (4). pp. 26:1-26:22.

Виды зависимостей

- **Потоковая (истинная) зависимость («чтение после записи», read-after-write, RAW)**

```
    for (int i = 0; i < N; i++)
    {
S1:     A[i] = B[i] + C[i];
S2:     D[i] = A[i];
    }
```

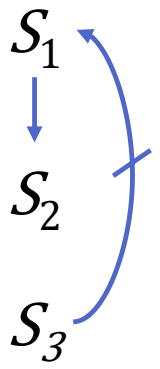
$S_1 \delta_= S_2$

- **Антизависимость («запись после чтения», write-after-read, WAR)**

```
    for (int i = 0; i < N; i++)
    {
S1:     A[i] = B[i] + C[i];
S2:     D[i] = A[i];
S3:     B[i] = D[i];
    }
```

$S_1 \delta_= S_2$

$S_3 \overline{\delta}_= S_1$



- **Выходная зависимость («запись после записи», write-after-write, WAW)**

```
    for (int i = 0; i < N; i++)
    {
S1:     A[i] = B[i] + C[i];
S2:     A[i+1] = A[i] + D[i];
    }
```

$S_1 \delta_= S_2$

$S_2 \delta^o_> S_1$