Семинар 1 (23)
Программирование сопроцессора Intel Xeon Phi

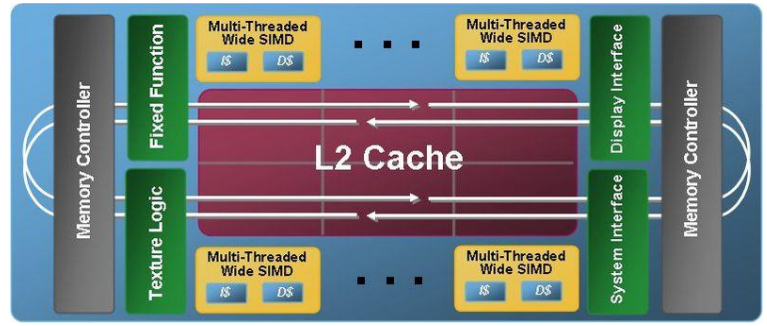
Михаил Курносов

E-mail: mkurnosov@gmail.com
WWW: www.mkurnosov.net

Цикл семинаров «Основы параллельного программирования»
Институт физики полупроводников им. А. В. Ржанова СО РАН
Новосибирск, 2016

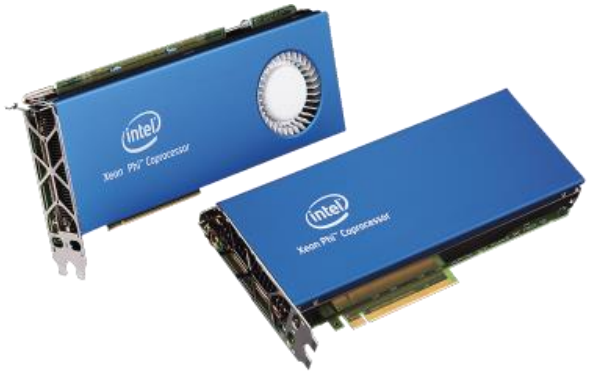
Intel Xeon Phi

- **Intel Teraflops Research Chip** (Polaris, 200x-2007 гг.)
  80 cores, 80 routers, self-correction system
- **Single-Chip Cloud Computer (SCC, 2009)**
  48 P54C Pentium cores, 4x6 2D-mesh of tiles (2 cores)
- **Intel Larrabee (200X-2010 гг.)**
  GPGPU: x86, cache coherency, 1024-bit ring bus, 4-way SMT,
  разработка отменена в 2010 г.



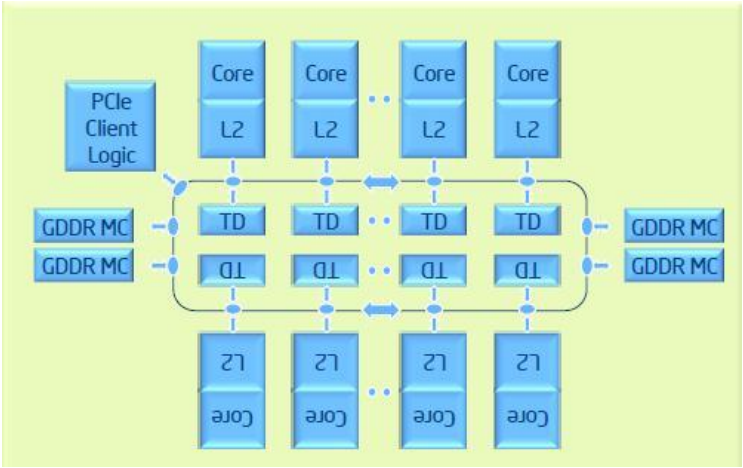
Larrabee GPU
(SIGGRAPH, 2008)

- **Intel MIC (Intel Many Integrated Core Architecture)**
  кеш-когерентный мультипроцессор с общей памятью,
  аппаратная многопоточность (4-way SMT),
  широкие векторные регистры (512 бит)
  - **Knights Ferry** (2010): 32 in-order cores, 4-way SMT, ring bus, 750 GFLOPS
  - **Knights Corner/ Xeon Phi** (2011): 22 nm, >= 50 cores
  - **Knights Landing** (2nd gen., 14 nm, 2013): 74 Airmont (Atom) cores, bootable
  - **Knights Hill** (3rd gen., 10 nm)

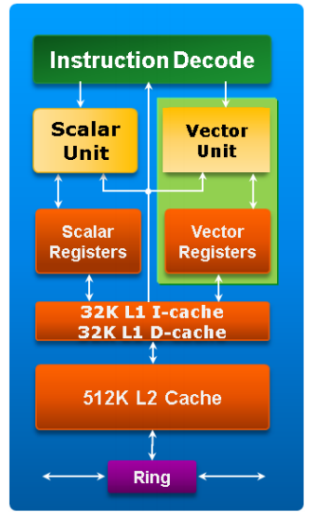


Intel Knights Corner micro-architecture

- **Больше 50 ядер Pentium P54C**:
  - x86 ISA, in-order, 4-way SMT, 512-bit SIMD units
  - Cache: 32 KB L1 data/i-cache, coherent L2 cache (512 KB), двусторонняя кольцевая шина
- Кольцевая шина: двусторонняя
- Подключается через шину PCI Express



- **cnmin (oak): Intel Xeon Phi 3120A**
  - **Cores:** 57 4-way SMT (core #57 for OS only!): 224 threads
  - **RAM**: GDDR5 6 GiB
  - **Intel Compiler:**
    ```
    $ source /opt/intel_cluster/bin/iccvars.sh intel64
    ```



https://software.intel.com/sites/default/files/managed/ee/4e/intel-xeon-phi-coprocessor-quick-start-developers-guide.pdf

https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner

Подсчет простых чисел (serial version)

```
int is_prime_number(int n)
{
    int limit = sqrt(n) + 1;
    for (int i = 2; i <= limit; i++) {
        if (n % i == 0) return 0;
    }
    return (n > 1) ? 1 : 0;
}

int count_prime_numbers(int a, int b)
{
    int nprimes = 0;
    if (a <= 2) {           /* Count '2' as a prime number */
        nprimes = 1;
        a = 2;
    }
    if (a % 2 == 0)         /* Shift 'a' to odd number */
        a++;

    /* Loop over odd numbers: a, a + 2, a + 4, ... , b */
    for (int i = a; i <= b; i += 2) {
        if (is_prime_number(i))
            nprimes++;
    }
    return nprimes;
}
```

Число операций $O(\sqrt{n})$

Проверка всех нечетных чисел в интервале $[a, b]$

Подсчет простых чисел (OpenMP)

```
int count_prime_numbers_omp(int a, int b)
{
    int nprimes = 0;
    /* Count '2' as a prime number */
    if (a <= 2) {
        nprimes = 1;
        a = 2;
    }
    /* Shift 'a' to odd number */
    if (a % 2 == 0)
        a++;

    #pragma omp parallel
    {
        /* Loop over odd numbers: a, a + 2, a + 4, ... , b */
        #pragma omp for schedule(dynamic, 100) reduction(+:nprimes)
        for (int i = a; i <= b; i += 2) {
            if (is_prime_number(i))
                nprimes++;
        }
    }
    return nprimes;
}
```

Подсчет простых чисел (OpenMP)

```
int count_prime_numbers_omp(int a, int b)
{
    int nprimes = 0;
    /* Count '2' as a prime number */
    if (a <= 2) {
        nprimes = 1;
        a = 2;
    }
    /* Shift 'a' to odd number */
    if (a % 2 == 0)
        a++;

    #pragma omp parallel
    {
        /* Loop over odd numbers: a, a + 2, a + 4, ... , b */
        #pragma omp for schedule(dynamic, 100) reduction(+:nprimes)
        for (int i = a; i <= b; i += 2) {
            if (is_prime_number(i))
                nprimes++;
        }
    }
    return nprimes;
}
```

cnmic (oak.cpct.sibsutis.ru)

System board: ASUS Z10PE-D8 WS (Dual CPU)
CPU: 2 x Intel Xeon E5-2620v3 (2.40GHz, Haswell, 6 cores)
RAM: 64 GiB, DDR4 2133 Mhz (4x8 GiB, 4x8 GiB)
Coprocessor: Intel Xeon Phi 3120A (Cores: 56 4-way SMT, RAM: GDDR5 6 GiB)

```
$ icc -std=c99 -Wall -O2 -fopenmp -c primes.c -o primes.o
$ icc -o primes primes.o -lm -fopenmp
```

```
$ export OMP_NUM_THREADS=12
$ ./primes
Count prime numbers in [1, 3000000]
Result (host serial): 216816
Result (host parallel): 216816
Execution time (host serial): 1.213376
Execution time (host parallel): 0.103723
Speedup host_serial/host_omp: 11.70
```

Подсчет простых чисел: Xeon Phi offload (serial)

```
#include <offload.h>

double run_phi_serial()
{
    #ifdef __INTEL_OFFLOAD
    printf("Intel Xeon Phi devices: %d\n", _Offload_number_of_devices());
    #endif

    int n;
    double t = wtime();
    #pragma offload target(mic) out(n)
    {
        n = count_prime_numbers_phi(a, b);
    }
    t = wtime() - t;

    printf("Result (phi serial): %d\n", n);
    return t;
}
```

Структурный блок *выгружается* (offload)
для выполнения на сопроцессор

Требуется указать:

- объекты, которые необходимо скопировать в память сопроцессора перед выполнением блока (**in**)
- объекты, которые необходимо скопировать из памяти сопроцессора после выполнением блока (**out**)

Подсчет простых чисел: Xeon Phi offload (serial)

```
__attribute__((target(mic))) int count_prime_numbers_phi(int a, int b)
{
    int nprimes = 0;

    /* Count '2' as a prime number */
    if (a <= 2) {
        nprimes = 1;
        a = 2;
    }

    /* Shift 'a' to odd number */
    if (a % 2 == 0)
        a++;

    /* Loop over odd numbers: a, a + 2, a + 4, ... , b */
    for (int i = a; i <= b; i += 2) {
        if (is_prime_number(i))
            nprimes++;
    }
    return nprimes;
}
```

Функции и переменные, доступ к которым осуществляется на сопроцессоре, помечаются атрибутом

`__attribute__((target(mic)))`

(гетерогенная компиляция)

Подсчет простых чисел: Xeon Phi offload (serial)

```
__attribute__((target(mic))) int is_prime_number(int n)
{
    int limit = sqrt(n) + 1;
    for (int i = 2; i <= limit; i++) {
        if (n % i == 0)
            return 0;
    }
    return (n > 1) ? 1 : 0;
}
```

Функции и переменные, доступ к которым осуществляется на сопроцессоре, помечаются атрибутом

`__attribute__((target(mic)))`

(гетерогенная компиляция)

Подсчет простых чисел: Xeon Phi offload (serial)

```
int main(int argc, char **argv)
{
    printf("Count prime numbers in [%d, %d]\n", a, b);
    double thost_serial = run_host_serial();
    double thost_par = run_host_parallel();
    double tphi_serial = run_phi_serial();

    printf("Execution time (host serial): %.6f\n", thost_serial);
    printf("Execution time (host parallel): %.6f\n", thost_par);
    printf("Execution time (phi serial): %.6f\n", tphi_serial);
    printf("Ratio phi_serial/host_serial: %.2f\n", tphi_serial / thost_serial);
    printf("Speedup host_serial/host_omp: %.2f\n", thost_serial / thost_par);

    return 0;
}
```

```
$ export OMP_NUM_THREADS=12
$ ./primes
Count prime numbers in [1, 3000000]
Result (host serial): 216816
Result (host parallel): 216816
Intel Xeon Phi devices: 1
Result (phi serial): 216816
Execution time (host serial): 1.213118
Execution time (host parallel): 0.103338
Execution time (phi serial): 18.031396
Ratio phi_serial/host_serial: 14.86
Speedup host_serial/host_omp: 11.74
```

На данном тесте ядро Intel Xeon Phi 3120A
в ~15 раз медленнее ядра Intel Xeon E5-2620 v3

Подсчет простых чисел: Xeon Phi offload (OpenMP)

```
double run_phi_parallel()
{
    #ifdef __INTEL_OFFLOAD
    printf("Intel Xeon Phi devices: %d\n", _Offload_number_of_devices());
    #endif

    int n;
    double t = wtime();
    #pragma offload target(mic) out(n)
    n = count_prime_numbers_phi_omp(a, b);
    t = wtime() - t;

    printf("Result (phi parallel): %d\n", n);
    return t;
}
```

Подсчет простых чисел: Xeon Phi offload (OpenMP)

```
__attribute__((target(mic))) int count_prime_numbers_phi_omp(int a, int b)
{
    int nprimes = 0;
    /* Count '2' as a prime number */
    if (a <= 2) {
        nprimes = 1;
        a = 2;
    }
    /* Shift 'a' to odd number */
    if (a % 2 == 0)
        a++;

    /* Loop over odd numbers: a, a + 2, a + 4, ... , b */
    #pragma omp parallel
    {
        #pragma omp for schedule(dynamic, 100) reduction(+:nprimes)
        for (int i = a; i <= b; i += 2) {
            if (is_prime_number(i))
                nprimes++;
        }
    }
    return nprimes;
}
```

Подсчет простых чисел: Xeon Phi offload (OpenMP)

```
int main(int argc, char **argv)
{
    printf("Count prime numbers in [%d, %d]\n", a, b);
    double thost_serial = run_host_serial();
    double thost_par = run_host_parallel();
    double tphi_serial = run_phi_serial();
    double tphi_par = run_phi_parallel();

    printf("Execution time (host serial): %.6f\n", thost_serial);
    printf("Execution time (host parallel): %.6f\n", thost_par);
    printf("Execution time (phi serial): %.6f\n", tphi_serial);
    printf("Execution time (phi parallel): %.6f\n", tphi_par);
    printf("Ratio phi_serial/host_serial: %.2f\n", tphi_serial / thost_serial);
    printf("Speedup host_serial/host_omp: %.2f\n", thost_serial / thost_par);
    printf("Speedup host_omp/phi_omp: %.2f\n", thost_par / tphi_par);
    printf("Speedup host_serial/phi_omp: %.2f\n", thost_serial / tphi_par);
    printf("Speedup phi_serial/phi_omp: %.2f\n", tphi_serial / tphi_par);

    return 0;
}
```

Подсчет простых чисел: Xeon Phi offload (OpenMP)

```
int main(int argc, char **argv)
{
    printf("Count prime numbers in [%d, %d]\n", a, b);
    double thost_serial = run_host_serial();
    double thost_par = run_host_parallel();
    double tphi_serial = run_phi_serial();
    double tphi_par = run_phi_parallel();

    printf("Execution time (host serial): %.6f\n", thost_serial);
    printf("Execution time (host parallel): %.6f\n", thost_par);
    printf("Execution time (phi serial): %.6f\n", tphi_serial);
    printf("Execution time (phi parallel): %.6f\n", tphi_par);
    printf("Ratio phi_serial/host_serial: %.2f\n", tphi_serial / thost_serial);
    printf("Speedup host_serial/host_omp: %.2f\n", thost_serial / thost_par);
    printf("Speedup host_omp/phi_omp: %.2f\n", thost_par / tphi_par);
    printf("Speedup host_serial/phi_omp: %.2f\n", thost_serial / tphi_pa
    printf("Speedup phi_serial/phi_omp: %.2f\n", tphi_serial / tphi_par);

    return 0;
}
```

```
$ cat ./launch.sh
#!/bin/sh

# Host OpenMP
export OMP_NUM_THREADS=12

# Xeon Phi OpenMP
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=224

./primes
```

```
$ ./launch.sh
Count prime numbers in [1, 3000000]
Execution time (host serial): 1.213278
Execution time (host parallel): 0.104636
Execution time (phi serial): 18.021162
Execution time (phi parallel): 0.770084
Ratio phi_serial/host_serial: 14.85
Speedup host_serial/host_omp: 11.60
Speedup host_omp/phi_omp: 0.14
Speedup host_serial/phi_omp: 1.58
Speedup phi_serial/phi_omp: 23.40
```

- Phi-OpenMP-версия на 224 потоках в 23 раза быстрее последовательной Phi-версии
- Phi-OpenMP-версия медленее Host-OpenMP-версии в ~7 раз

Xeon Phi: привязка потоков к ядрам (affinity)

```
$ cat ./launch.sh
#!/bin/sh

# Host OpenMP
export OMP_NUM_THREADS=12

# Xeon Phi OpenMP
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=56
export MIC_KMP_AFFINITY=verbose

./primes
```

```
Execution time (host serial): 1.213139
Execution time (host parallel): 0.108063
Execution time (phi serial): 18.027308
Execution time (phi parallel): 0.541261
Ratio phi_serial/host_serial: 14.86
Speedup host_serial/host_omp: 11.23
Speedup host_omp/phi_omp: 0.20
Speedup host_serial/phi_omp: 2.24
Speedup phi_serial/phi_omp: 33.31
```

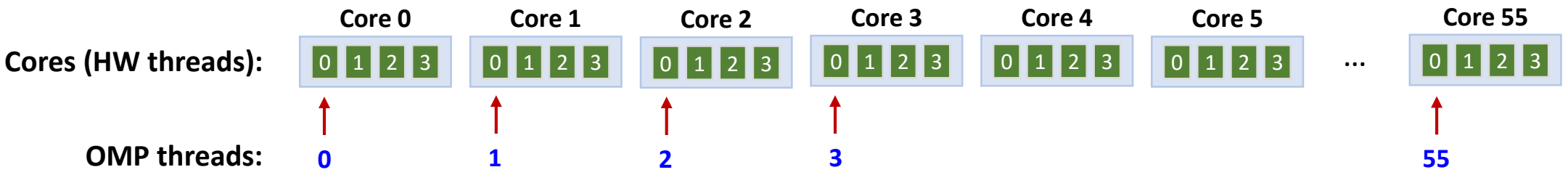
```
OMP: Info #204: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #205: KMP_AFFINITY: cpuid leaf 11 not supported - decoding legacy APIC ids.
OMP: Info #149: KMP_AFFINITY: Affinity capable, using global cpuid info
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected:
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44
,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85
,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,
120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150
,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,18
1,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,2
12,213,214,215,216,217,218,219,220,221,222,223,224}
OMP: Info #156: KMP_AFFINITY: 224 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #159: KMP_AFFINITY: 1 packages x 56 cores/pkg x 4 threads/core (56 total cores)
```

Xeon Phi: привязка потоков к ядрам (affinity)

```
OMP: Info #206: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 0 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 0 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 0 thread 2
OMP: Info #171: KMP_AFFINITY: OS proc 4 maps to package 0 core 0 thread 3
OMP: Info #171: KMP_AFFINITY: OS proc 5 maps to package 0 core 1 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 6 maps to package 0 core 1 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 7 maps to package 0 core 1 thread 2
OMP: Info #171: KMP_AFFINITY: OS proc 8 maps to package 0 core 1 thread 3
OMP: Info #171: KMP_AFFINITY: OS proc 9 maps to package 0 core 2 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 10 maps to package 0 core 2 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 11 maps to package 0 core 2 thread 2
OMP: Info #171: KMP_AFFINITY: OS proc 12 maps to package 0 core 2 thread 3
...
OMP: Info #171: KMP_AFFINITY: OS proc 221 maps to package 0 core 55 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 222 maps to package 0 core 55 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 223 maps to package 0 core 55 thread 2
OMP: Info #171: KMP_AFFINITY: OS proc 224 maps to package 0 core 55 thread 3
```

Xeon Phi: привязка потоков к ядрам (affinity)

```
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 0 bound to OS proc set {1}
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 1 bound to OS proc set {5}
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 2 bound to OS proc set {9}
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 3 bound to OS proc set {13}
...
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 54 bound to OS proc set {217}
OMP: Info #242: KMP_AFFINITY: pid 12242 thread 55 bound to OS proc set {221}
```



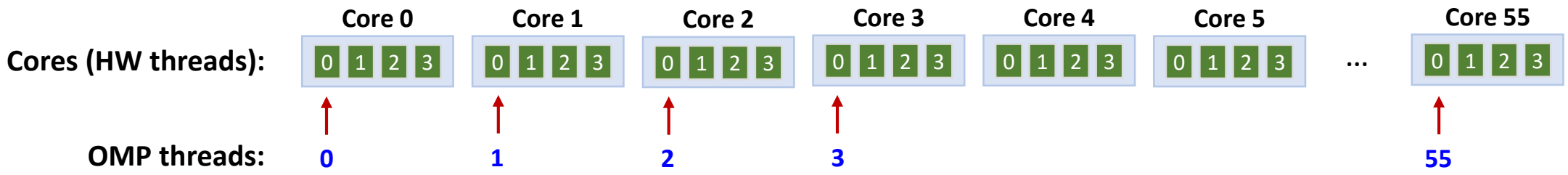
Default Affinity

Xeon Phi: привязка потоков к ядрам (affinity)

```
$ cat ./launch.sh
#!/bin/sh

# Host OpenMP
export OMP_NUM_THREADS=12

# Xeon Phi OpenMP
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=56
export MIC_KMP_AFFINITY=granularity=fine,balanced

./primes
```

```
Execution time (host serial): 1.213125
Execution time (host parallel): 0.103671
Execution time (phi serial): 18.028928
Execution time (phi parallel): 0.482872
Ratio phi_serial/host_serial: 14.86
Speedup host_serial/host_omp: 11.70
Speedup host_omp/phi_omp: 0.21
Speedup host_serial/phi_omp: 2.51
Speedup phi_serial/phi_omp: 37.34
```

fine, balanced

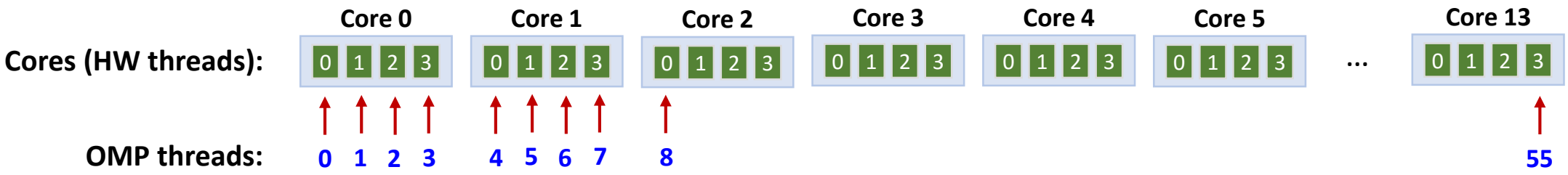


Xeon Phi: привязка потоков к ядрам (affinity)

```
$ cat ./launch.sh
#!/bin/sh

# Host OpenMP
export OMP_NUM_THREADS=12

# Xeon Phi OpenMP
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=56
export MIC_KMP_AFFINITY=granularity=fine,compact
./primes
```

```
Execution time (host serial): 1.213115
Execution time (host parallel): 0.103378
Execution time (phi serial): 18.030468
Execution time (phi parallel): 1.478703
Ratio phi_serial/host_serial: 14.86
Speedup host_serial/host_omp: 11.73
Speedup host_omp/phi_omp: 0.07
Speedup host_serial/phi_omp: 0.82
Speedup phi_serial/phi_omp: 12.19
```

fine, compact

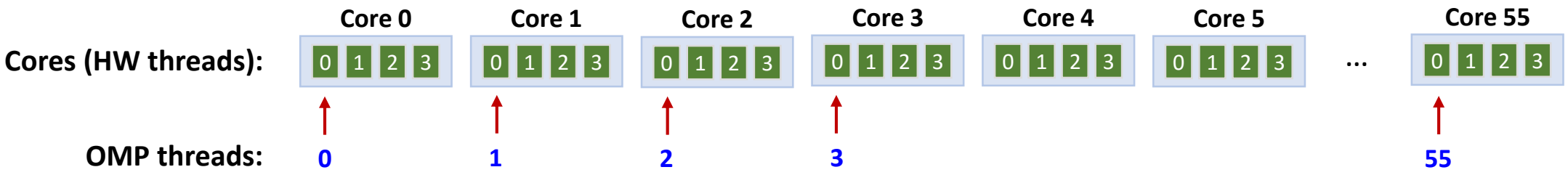


Xeon Phi: привязка потоков к ядрам (affinity)

```
$ cat ./launch.sh
#!/bin/sh

# Host OpenMP
export OMP_NUM_THREADS=12

# Xeon Phi OpenMP
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=56
export MIC_KMP_AFFINITY=granularity=fine,scatter
./primes
```

```
Execution time (host serial): 1.213140
Execution time (host parallel): 0.108781
Execution time (phi serial): 18.024953
Execution time (phi parallel): 0.551100
Ratio phi_serial/host_serial: 14.86
Speedup host_serial/host_omp: 11.15
Speedup host_omp/phi_omp: 0.20
Speedup host_serial/phi_omp: 2.20
Speedup phi_serial/phi_omp: 32.71
```

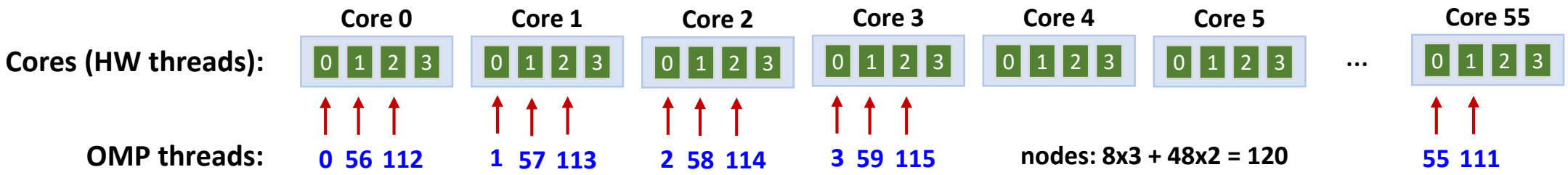
fine, scatter



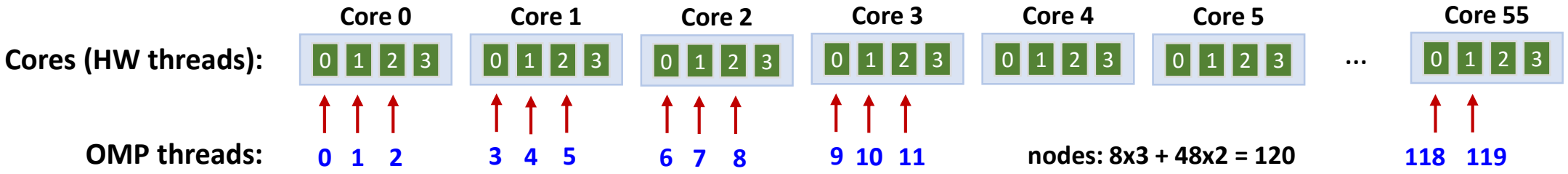
Xeon Phi: привязка потоков к ядрам (affinity)

NUM_THREADS = 120

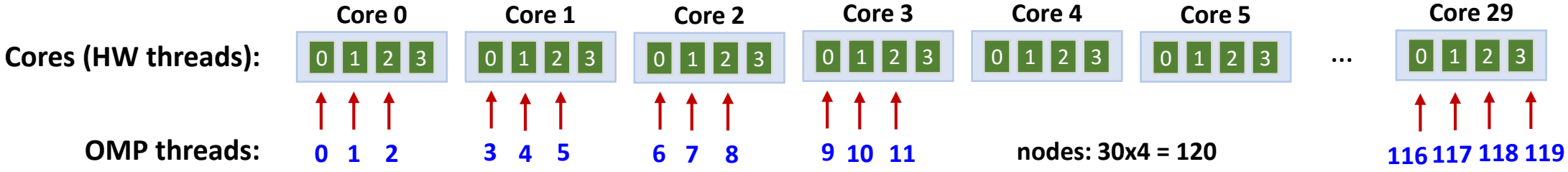
fine, scatter: core_id = thread_id % ncores



fine, balanced:



fine, compact:



Задание

- Реализуйте вычисление определенного интеграла методом прямоугольников на Intel Xeon Phi (шаблон размещен в каталоге _task_integrate):
  1. Выгрузите выполнение функции integrate_phi_omp на сопроцессор
  2. Функцию integare_phi_omp реализуйте на OpenMP