Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Программирование для Intel Xeon Phi

Лабораторная работа
Компиляция и запуск приложений на Intel Xeon Phi

Горшков А.В.
Линев А.В.
Сиднев А.А.

Архангельск, 2014

Содержание

Введение

- В лабораторной работе рассматриваются модели программирования на сопроцессоре Intel Xeon Phi
- Intel Xeon Phi – новый сопроцессор от компании Intel, призванный существенно ускорить процесс вычислений для некоторого класса задач, алгоритмы решения которых допускают существенную степень параллелизма и векторизации
- Сопроцессор основан на архитектуре Intel Many Integrated Core (MIC), содержит несколько десятков x86 CPU ядер, поддерживает сотни потоков исполнения

Цели работы

- **Изучение режимов и способов компиляции и запуска программ на Intel Xeon Phi:**
  - Изучение моделей программирования на Intel Xeon Phi
  - Освоение способов компиляции и запуска программ на одном или нескольких сопроцессорах применительно к каждой из рассматриваемых моделей программирования

Тестовая инфраструктура

Процессор	Intel Xeon E5-2690 (2.9 GHz, 8 ядер)
Сопроцессор	Intel Xeon Phi 7110X (61 ядро, 244 потока)
Память	64 Gb
Операционная система	Linux CentOS 6.2
Компилятор, профилировщик, отладчик	Intel C/C++ Compiler 13

Тестовая инфраструктура

- URL login.tornado.hpc.susu.ac.ru
- Логины
  - Ауд.209: **fpk1 – fpk9**
  - Ауд.301: **fpk10 – fpk20**
- Пароль: **intelxeonphi2014**
- При запросе ресурсов необходимо указывать дополнительный ключ – имя резервации:
  - sbatch **--reservation=scc_phi_traininig**
  - salloc **--reservation=scc_phi_traininig**

Используемое ПО

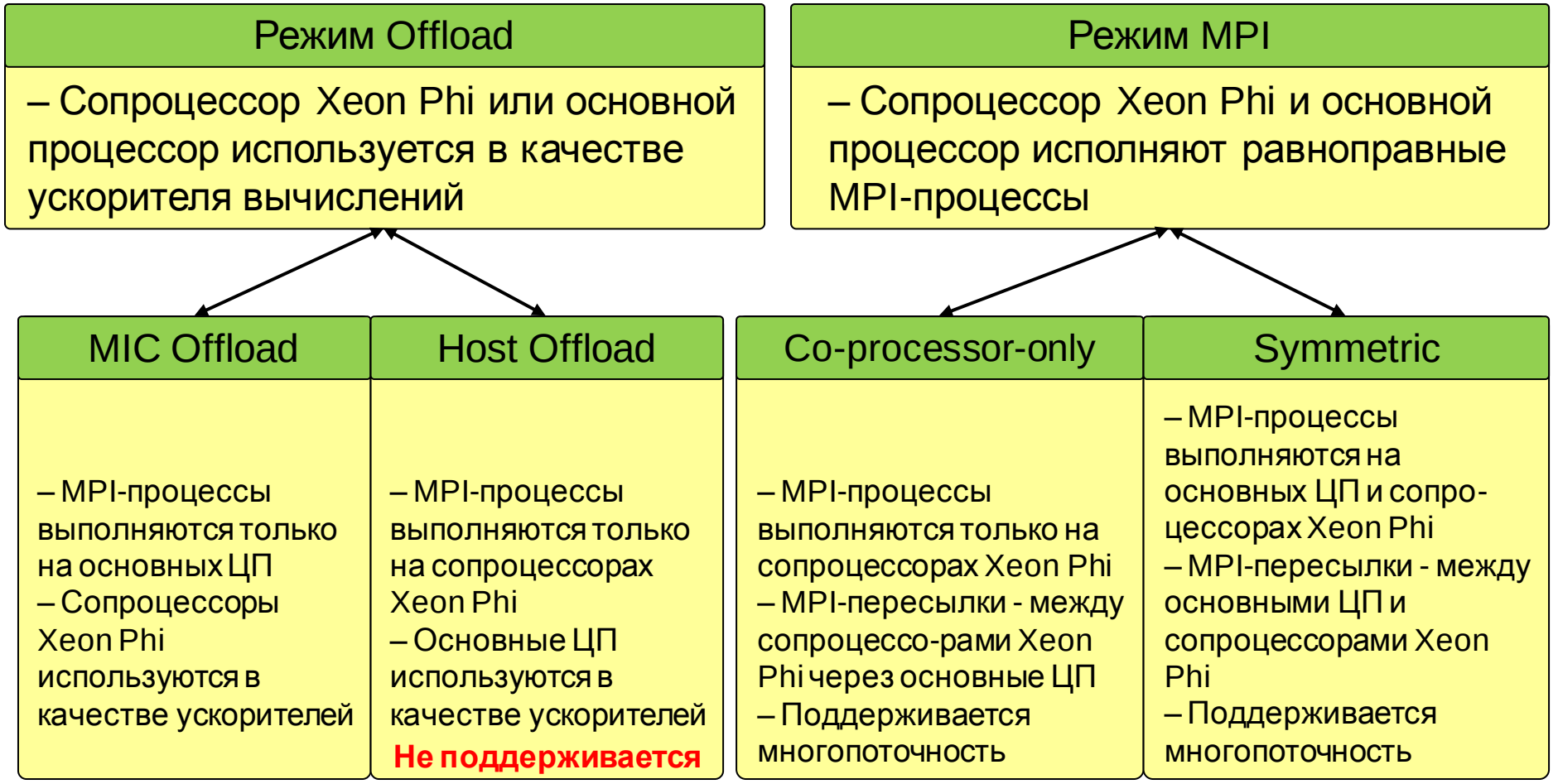
- putty – ssh-клиент для доступа к узлам кластера
- WinSCP – графический клиент SCP/FTP/SFTP

- SLURM (Simple Linux Utility for Resource Management) – система управления кластером
  - salloc – резервирование узлов кластера
  - sbatch – создание задания с использованием скрипта
  - scancel – отправка сигнала заданию
  - sinfo – просмотр информации об узлах кластера
  - squeue – просмотр информации о текущих заданиях
  - srun – запуск задания на выполнение

Используемое ПО

- Hydra – система управления процессами, использующаяся для запуска параллельных заданий
  - mpiexec.hydra – запуск параллельной MPI-программы
  - native_run.sh – служебный скрипт для запуска MPI-приложений с моделью использования только сопроцессора
  - symmetric_run.sh – служебный скрипт для запуска MPI-приложений с моделью симметричного выполнения

Модели программирования приложений



Режим offload

- **Offload модель** предполагает использование Xeon Phi в режиме сопроцессора, т.е. дополнительного вычислительного устройства, доступ к которому осуществляется с помощью специальных команд из кода, выполняемого на обычном центральном процессоре
- Поддерживаются языки программирования C/C++ и Fortran
- При выполнении MPI программы в этом режиме, ранги присваиваются только центральным процессорам
- Поддерживается библиотекой Intel MPI, начиная с версии 4.0 update 3, для операционных систем семейства Linux при условии, что в качестве центрального процессора выступает процессор семейства Intel Xeon

Режим offload: вывод максимального числа потоков процессора и сопроцессора…

- Изучите код исходного файла **main01.cpp**, который находится в папке **/tmp/xeonphilabs/lab1_1_thread_test**

```
#pragma offload_attribute(push, target(mic))
void testThreadCount(){
    int thread_count;
    #pragma omp parallel
    {
        #pragma omp single
            thread_count = omp_get_num_threads();
    }
    printf("Number of threads: %d\n", thread_count);
}
#pragma offload_attribute(pop)
```

Режим offload: вывод максимального числа потоков сопроцессора…

- Если работа ведется на кластере с системой управления SLURM, то перед запуском программы необходимо выделить себе сопроцессор для монопольного использования:

```
salloc –N 1 --gres=mic:1
```

- Ключ «-N 1» означает, что нужен один узел кластера.
- Ключ «--gres=mic:1» говорит о том, что требуется узел как минимум с одним сопроцессором.

```
-sh-4.1$ salloc -N 1 --gres=mic:1
salloc: Pending job allocation 9180
salloc: job 9180 queued and waiting for resources
salloc: job 9180 has been allocated resources
salloc: Granted job allocation 9180
```

На Tornado может понадобиться явно указать, что нужно использовать узлы из заранее заданного для нас резерва (ключ «--reservation= scc_phi_training»)

Режим offload: вывод максимального числа потоков сопроцессора…

- Скомпилируйте файл с помощью Intel Compiler:

```
icc -02 -openmp main01.cpp –o lab1_thread_test
```

- Для запуска программы с использованием сопроцессора нужно выполнить команду `mpiexec.hydra`:

```
mpiexec.hydra –perhost 1 ./lab1_thread_test
```

- Команда запустит параллельную программу на узлах, выделенных ранее с помощью команды `salloc`
- Ключ «–perhost 1» говорит компилятору, что нужно запустить программу в 1 процесс

Режим offload: вывод максимального числа потоков сопроцессора…

- В случае, если свободных узлов с заданными параметрами нет, команда будет ожидать их появления, блокируя консоль
- Если доступ предоставлен, то будет создана отдельная задача в системе управления кластером, номер задачи будет указан в качестве результата выполнения команды.
- Монопольный доступ будет предоставлен в течение некоторого промежутка времени (определяется системой управления кластером).
- Освободить сопроцессор можно заранее командой:

```
scancel <Номер задачи>
```

- Посмотреть состояние очереди можно командой **squeue**

Режим offload: вывод максимального числа потоков сопроцессора…

- Результаты запуска программы приведены на рисунке:

```
[unn-kozinov@login lab1_thread_test]$ mpiexec.hydra -perhost 1 ./lab1_thread_tes
t
Number of threads: 240
Intel CPU:
Number of threads: 12
Intel Xeon Phi:
Number of coprocessors: 1
```

Режим offload: вывод максимального числа потоков сопроцессора…

- Еще один способ запуска offload программы – запуск в пакетном режиме (не требует предварительного выделения сопроцессора в монопольное пользование с помощью команды salloc)
- Необходимо написать скрипт запуска (напр., **./run01.sh**):

```
#!/bin/sh
mpiexec.hydra -perhost X -n X*Y ./program_name
#mpiexec.hydra -perhost 1 -n 1 ./lab1_thread_test
```

- Ключ «-perhost X» говорит о запуске задачи в X процессов на узел.
- Ключ «-n X*Y» говорит о запуска задачи в X*Y процессов, где Y – требуемое число узлов кластера

Режим offload: вывод максимального числа потоков сопроцессора…

- Для постановки задачи в очередь нужно выполнить команды:

```
module load launcher/intel
sbatch –N Y --gres=mic:1 ./run01.sh
```

- ***Для запуска на Tornado*** может потребоваться:

```
sbatch –N 1 --gres=mic:1 --reservation=scc_phi_training
./run01.sh
```

- Первая команда подгружает необходимые для запуска библиотеки Intel
- Вторая команда ставит задачу в очередь. Ключ «-N Y» означает, что для запуска будут использоваться Y узлов кластера (мы будем использовать -N 1)
- sbatch выведет номер задачи в очереди на исполнение

Режим offload: вывод максимального числа потоков сопроцессора

- Результаты запуска будут записаны в текущую директорию, консольный вывод будет содержаться в файле **slurm-<Номер задачи>.out**
- Для просмотра консольного вывода можно воспользоваться командой

```
cat slurm-<Номер задачи>.out
```

- **Задание:** выполните запуск программы для расчета скалярного произведения в пакетном режиме

Режим offload: скалярное произведение

- Изучите код исходного файла **main02.cpp**, который находится в папке **/tmp/xeonphilabs/lab1_2_dot_offload**.
- Скомпилируйте файл с помощью Intel Compiler:

```
icc -O2 -openmp main02.cpp -o lab1_dot_offload
```

- Процесс запуска программы аналогичен предыдущему
  - написать скрипт запуска run02.sh

```
#!/bin/sh
mpiexec.hydra -perhost X -n X*Y ./program_name
```

  - sbatch -N 1 --gres=mic:1 ./run02.sh

```
[linev@login lab1_2_dot_offload]$ icc -O2 -openmp -o main02 main02.cpp
[linev@login lab1_2_dot_offload]$ sbatch -N 1 --gres=mic:1 ./run02.sh
Submitted batch job 25553
[linev@login lab1_2_dot_offload]$ cat slurm-25553.out
Host name: node471
CPU dot: 27.259979
MIC dot: 27.259979
```

Режим исполнения на сопроцессоре

- Программа исполняется только на сопроцессоре (одном или многих) без участия центрального процессора
- Поддерживаются языки программирования C/C++ и Fortran
- При выполнении MPI программы в этом режиме, ранги присваиваются только сопроцессорам

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Изучите код исходного файла **main03.cpp**, который находится в папке **/tmp/xeonphilabs/lab1_3_dot_native**. По сути, это обычная параллельная OpenMP программа.
- Скомпилируйте файл для выполнения на сопроцессоре с помощью Intel Compiler:

```
icc -02 –openmp -mmic main03.cpp –o lab1_dot_native.mic
```

- Обратите внимание на ключ «-mmic», который говорит о том, что код компилируется именно под сопроцессор
- К имени программы для Intel Xeon Phi рекомендуется добавлять расширение «.mic». При работе с кластером оно может быть обязательно.

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Запуск приложения на сопроцессоре можно выполнить тремя способами
- Способ 1
  - Зарезервировать узел с помощью salloc; зайти на выделенный узел с помощью ssh; скопировать файл на сопроцессор; зайти на Xeon Phi с помощью ssh; запустить программу из командной строки
    - `salloc -N 1 --gres=mic:1 --reservation=scc_phi_training`
    - `ssh <nodename>`
    - `cp <progname> <xeon_phi_dir/>`
    - `ssh mic0`
    - `xeon_phi_dir/progname`

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Способ 2
  - Зарезервировать узел с помощью salloc; зайти на выделенный узел с помощью ssh; запустить программу с помощью mpiexec.hydra
    - `salloc -N 1 --gres=mic:1 --reservation=scc_phi_training`
    - `ssh <nodename>`
    - `mpiexec.hydra -host mic0 -n 1 -perhost 1 ./lab1_dot_native.mic`

Ключ «-host mic0» говорит о запуске на сопроцессоре. «mic0» - имя этого сопроцессора как отдельного узла кластера.

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Способ 3
- Если работа ведется с системой управления кластером SLURM, то запуск можно выполнить через очередь задач:

```
sbatch -N 1 --gres=mic:2 --reservation=scc_phi_training
native_run.sh ./lab1_dot_native
```

- Обратите внимание, что имя исполняемого файла задается без расширения «.mic», хотя само это расширение у файла должно присутствовать.
- Ключ «-N» задает число узлов, на котором будет выполнена программа.
- Ключ «--gres=mic:2» говорит о необходимости выделения узлов минимум с двумя сопроцессорами Intel Xeon Phi.
- Иногда может потребоваться задание полного пути до исполняемого файла (***например, при запуске на Tornado***).

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Скрипт «native_run.sh» предназначен для запуска программ в режиме исполнения на сопроцессоре. Это встроенный скрипт системы управления, но по умолчанию он не виден. Для его подключения нужно выполнить команду

```
module load launcher/mic
```

- Данная команда позволяет управлять набором модулей, которые используются системой управления кластером в данный момент.
- Модуль `launcher/mic` конфликтует с модулем `launcher/intel`, который таким образом нужно предварительно выгрузить

```
module unload launcher/intel
```

- Список всех модулей можно получить командой

```
module avail
```

Режим исполнения на сопроцессоре: скалярное произведение на одном Xeon Phi…

- Список используемых на данный момент модулей можно посмотреть с помощью команды

```
module list
```

- Результаты запуска будут записаны в текущую директорию, консольный вывод будет содержаться в файле **slurm-<Номер задачи>.out**
- Для просмотра консольного вывода можно воспользоваться командой

```
cat slurm-<Номер задачи>.out
```

```
-sh-4.1$ cat slurm-9197.out
Result: 27.259979
Result: 27.259979
```

Режим исполнения на сопроцессоре: скалярное произведение, MPI версия…

- При запуске программы в режиме исполнения на сопроцессоре количество создаваемых процессов на узел кластера по умолчанию может варьироваться.
- Это число обычно зависит от настроек системы управления каждого отдельного кластера.
- Число MIC-процессов на узел задается с помощью переменной окружения **MIC_PPN**. Для запуска программы с нужным числом процессов на каждый MIC следует выполнить команды:

```
export MIC_PPN=2
sbatch –N 1 --gres=mic:1 --reservation=scc_phi_training
native_run.sh ./lab1_dot_native
```

Режим исполнения на сопроцессоре: скалярное произведение, MPI версия…

- Изучите код исходного файла **main04.cpp**, который находится в папке **/tmp/xeonphilabs/lab1_4_dot_native_mpi**. Программа выполняет вычисление скалярного произведения набора векторов.
- Скомпилируйте файл для выполнения на сопроцессоре с помощью Intel Compiler:

```
mpicc -O2 -openmp -mmic main04.cpp -o ./lab1_dot_native_mpi.mic
```

- Обратите внимание на ключ «-mmic», который говорит о том, что код компилируется именно под сопроцессор.
- К имени программы для Intel Xeon Phi рекомендуется добавлять расширение «.mic». При работе с кластером оно может быть обязательно.

Режим исполнения на сопроцессоре: скалярное произведение, MPI версия…

- Команда запуска программы с основного (процессорного) узла в этом случае будет иметь вид:

```
mpiexec.hydra -hosts 4 mic0 mic1 mic2 mic3 -n 4
-perhost 1 ./lab1_dot_native_mpi.mic
```

- Для запуска на кластере с системой управления SLURM нужно выполнить команду:

```
sbatch -N 2 --gres=mic:1 --reservation=scc_phi_training
native_run.sh ./lab1_dot_native_mpi
```

Режим исполнения на сопроцессоре: скалярное произведение, MPI версия

- Результаты работы программы:

```
-sh-4.1$ cat slurm-9377.out
List of nodes:
node60-mic0
node61-mic0
node60-mic1
node61-mic1
Results:
128.796753
126.939026
122.497299
128.270416
120.303009
129.170853
113.081268
128.496140
118.298569
119.484879
126.637024
121.023445
123.013031
134.524841
125.303116
114.874908
117.644920
122.401176
125.044220
115.545937
```

Симметричный режим

- Программа исполняется как на сопроцессорах, так и на центральных процессорах.
- Поддерживаются языки программирования C/C++ и Fortran.
- При выполнении MPI программы в этом режиме каждый процессор и сопроцессор имеет отдельный ранг.

Симметричный режим: скалярное произведение, MPI версия…

- Изучите код исходного файла **main05.cpp**, который находится в папке **/tmp/xeonphilabs/lab1_5_dot_symmetric**. Программа выполняет вычисление скалярного произведения набора векторов.
- Для запуска в симметричном режиме необходимо дважды выполнить компиляцию программы:
  - Для центрального процессора:

```
mpicc -O2 -openmp main05.cpp -o ./lab1_dot_symmetric
```

  - Для сопроцессора:

```
mpicc -O2 -openmp -mmic main05.cpp -o ./lab1_dot_symmetric.mic
```

Симметричный режим: скалярное произведение, MPI версия…

- Обратите внимание, что при компиляции для сопроцессора дополнительно используется ключ «-mmic»
- Также следует обратить внимание на то, что имена исполняемых файлов отличаются только на расширение «.mic». Такое именование обязательно при работе с системой управления SLURM.

Симметричный режим: скалярное произведение, MPI версия…

- Для запуска программы в симметричном режиме следует воспользоваться командой:

```
mpiexec.hydra -hosts 2 node0 node1 -n 2 -
perhost 1 ./lab1_dot_symmetric: \
-hosts 4 mic0 mic1 mic2 mic3 -n 4 -perhost 1 -
wdir /tmp /tmp/lab1_dot_symmetric.mic
```

- При работе на кластере с системой управления SLURM команда запуска будет такая:

```
sbatch -N 2 --gres=mic:1 --reservation=scc_phi_training
symmetric_run.sh ./lab1_dot_symmetric
```

Симметричный режим: скалярное произведение, MPI версия

- Скрипт «symmetric_run.sh» содержится в модуле «launcher/mic»:

```
module load launcher/mic
```

- Обратите внимание, что иногда необходимо указывать полный путь до исполняемого файла.

```
-sh-4.1$ cat slurm-9390.out
List of nodes:
Node node195
Node node195
Node node196
Node node196
Node node195-mic1
Node node195-mic0
Node node196-mic1
Node node196-mic0
Results:
128.796753
126.939026
122.497299
128.270416
120.303001
129.170837
113.081268
128.496140
118.298569
119.484879
126.637024
121.023445
123.013031
134.524841
125.303116
114.874908
117.644920
122.401176
125.044220
115.545937
```

Дополнительные задания

- Реализовать умножение матрицы на вектор в режиме Offload.
- Реализовать умножение матрицы на вектор в режиме работы только на сопроцессоре. Считать, что программа предназначена для выполнения на одном сопроцессоре.
- Реализовать умножение матрицы на вектор в симметричном режиме. Обеспечить два уровня параллелизма – одновременное выполнение умножения строк матрицы на вектор и параллельное вычисление скалярного произведения векторов.

Литература

- Intel Xeon Phi Coprocessor System Software Developers Guide, revision 2.03, 2012
- Best Known Methods for Using OpenMP on Intel Many Integrated Core (Intel MIC) Architecture, Volume 1a, January 29, 2013
- Green R.W. Effective Use of the Intel Compiler's Offload Features: [http://software.intel.com/en-us/articles/effective-use-of-the-intel-compilers-offload-features]

Авторский коллектив

- Горшков Антон Валерьевич,
  ассистент кафедры Математического обеспечения ЭВМ факультета ВМК ННГУ.
  anton.v.gorshkov@gmail.com