ИСПОЛЬЗОВАНИЕ МНОГОЯДЕРНЫХ УСКОРИТЕЛЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПРОПОЗИЦИОНАЛЬНОЙ ВЫПОЛНИМОСТИ

Кирюшин Н.К.[1], Михалев И.В.[2] Email: Kiryushin17104@scientifictext.ru

[1]Кирюшин Никита Константинович – магистрант;
[2]Михалев Илья Викторович – магистрант,
кафедра вычислительной техники,
Национальный исследовательский университет
Московский институт электронной техники,
г. Москва

Аннотация: *проведено экспериментальное исследование применимости различных многоядерных аппаратных ускорителей для решения задачи выполнимости булевых формул. Для проведения экспериментальных исследований разработаны решатели, учитывающие особенности исследуемых аппаратных платформ. В данной работе рассматривалось применение графических ускорителей и универсальных многоядерных ускорителей Intel Xeon Phi. Представленные результаты дают понятие об алгоритмах, применимых для решения задачи выполнимости булевых формул с использованием многоядерных аппаратных ускорителей, а также о группах задач, относящихся к задаче выполнимости булевых формул, для которых использование многоядерных ускорителей обосновано.*
Ключевые слова: *пропозициональная выполнимость, выполнимость булевых формул, многоядерные ускорители, графические ускорители, DPLL, SLS, GPU, CUDA, Xeon Phi.*

USING MANYCORE HARDWARE ACCELERATORS FOR BOOLEAN SATISFIABILITY SOLVING

Kiryushin N.K.[1], Mikhalev I.V.[2]

[1]Kiryushin Nikita Konstantinovich – Undergraduate;
[2]Mikhalev Ilya Viktorovich – Undergraduate,
COMPUTER ENGINEERING DEPARTMENT
NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOG,Y
MOSCOW

Abstract: *an experimental study of the applicability of various manycore hardware accelerators to solve Boolean satisfiability (SAT) problem is carried out. Solvers, taking into account features of researched hardware platforms are developed to be used in the experimental researches. In this paper, we considered the use of GPU and Intel Xeon Phi coprocessors. The presented results give an idea of algorithms that are applicable for solving the problem of satisfiability of Boolean formulas using multi-core hardware accelerators, as well as groups of problems related to the problem of satisfiability of Boolean formulas for which the use of manycore accelerators is justified.*
Keywords: *propositional satisfiability, Boolean satisfiability, manycore coprocessors, graphic processing unit, DPLL, SLS, GPU, CUDA, Xeon Phi.*

УДК 004.272.23

Введение

Задача выполнимости булевых формул (задача пропозициональной выполнимости; англ. Satisfiability Problem, SAT) является одной из фундаментальных NP-полных задач в информатике. Задача располагается на стыке логики, теории графов, теории вычислений и методов оптимизации. Простота формализации делает задачу идеальной для моделирования комплексных вычислительных проблем из многих прикладных областей, в том числе проверки аппаратных средств и автоматического доказательства теорем. Интерес к задаче не ослабевает также из-за возможности свести к SAT любую задачу из класса $\mathcal{NP}$. Несмотря на то, что в общем случае проблема выполнимости не разрешима за полиномиальное время, нахождение случаев, когда ответ может быть получен быстро, очень важно для различных прикладных задач.

В свете развития в последнее десятилетие платформ многоядерных ускорителей представляется важным установить, насколько хорошо вписываются их специфику алгоритмы решения SAT.

Используемые алгоритмы

Среди всех решателей SAT можно выделить две главные категории, наиболее широко используемые исследователями, это соответственно полные и неполные решатели SAT.

Полные решатели SAT

Полный решатель SAT это алгоритм, ставящий целью проверить выполнимость пропозициональной формулы. Полный решатель гарантирует получение результата проверки, вне зависимости, выполнима ли формула или нет. Подавляющее большинство современных полных решателей основывается на классическом алгоритме DPLL (1969). Несмотря на возраст, DPLL до сих пор является высокоэффективным алгоритмом решения SAT, даже по современным меркам производительности. Фундаментальными принципами алгоритма DPLL являются обратное отслеживание и парадигма разделения пространства поиска. Сначала алгоритм упрощает формулу, назначая значения некоторым переменным таким образом, что если часть проблема выполнима, то выполнима вся формула. Если назначение ведет к невыполнимости формулы, алгоритм назначает переменным противоположные значения и продолжает эту процедуру рекурсивно до тех пор, пока не найдено решение или пока не пройдено все пространство поиска.

Во время этого процесса DPLL активно использует две вспомогательные процедуры, т.н. распространение переменной и удаление чистых литералов, повышающие эффективность алгоритма [1]. Современные исследования в этой области в основном посвящены методам повышения эффективности этих вспомогательных процедур.

Распространение переменной. Некоторые дизъюнкты содержат только один литерал, переменной которого уже назначено значение. Такие дизъюнкты можно удалить из формулы, не влияя тем самым на поиск значений остальных переменных. Удаления из формулы могут повлечь каскады единичных дизъюнктов, в результате размер исходной формулы может быть радикально снижен.

Удаление чистых литералов. Если переменная входит в формулу только с одной полярностью (положительной или отрицательной), все дизъюнкты с этой переменной могут быть удалены из формулы, т.к. назначение значения «истина» этой переменной приведет к истинности всех дизъюнктов с ее участием.

Несмотря на то, что современные алгоритмы, основанные на DPLL, эффективнее оригинального алгоритма на порядки, недостаток способности алгоритма к паралеллизации делает их очень сложными для реализации для высоконагруженных параллельных сред. В последние годы были предложены принципиально другие полные алгоритмы решения SAT, например, алгоритм "слияния и проверки", однако большинству из них предстоят годы оптимизации.

Неполные решатели SAT

Неполный решатель SAT ставит целью проверить выполнимость формулы за отведенное ему время работы. Получение результата не гарантируется, кроме того неполные решатели не способны доказать невыполнимость формулы. Однако возможности полных решателей по быстрому поиску решений для некоторых довольно больших видов формул значительно компенсирует их недостатки.

Неполные решатели SAT по большей части основаны на алгоритме стохастического локального поиска и генетических алгоритмах, большинство из которых весьма подходят к специфике параллельных компьютерных архитектур. Это дает неполным решателям неоценимое преимущество в паралеллизации. Реализованный в рамках этого исследования решатель для GPU использует стратегию случайного перехода.

Стратегия случайного перехода. Типичный алгоритм SLS начинается с присвоения переменным случайных значений и последующего итерационного генерирования новых наборов значений путем обращения значения одной переменной. Последняя процедура называется обращением. Итерации идут до тех пор, пока не будет найдено решение либо достигнуто условие останова [2].

Различные варианты алгоритма SLS различают в первую очередь по методу выбора следующего набора значений переменных. В использованном алгоритме был имплементирован «жадный» метод, основанный принятии во внимание числа невыполненных дизъюнктов для выбора следующей обращаемой переменной. На каждой итерации алгоритм либо выбирает литерал из случайного невыполненного дизъюнкта (с вероятностью p), либо литерал с наименьшим числом невыполненных дизъюнктов (с вероятностью 1 – p). Вероятность случайного выбора была добавлена в алгоритм с целью избежать попадания в локальный минимум, когда обращение не приводит к лучшему, чем предыдущий набору значений.

Многоядерные ускорители

Использование аппаратных ускорителей для ускорения вычислений не является новой идеей, так, сопроцессоры для реализации операций с плавающей точкой появились в настольных компьютерах ещё в 70х годах XX века. Развитие технологии, распространение многоядерных процессоров и необходимость решения сложных вычислительных задач привели к тому, что современные аппаратные ускорители являются высокопроизводительными вычислительными средствами со специфической архитектурой, предназначенными для эффективного решения определённых классов задач. Также, разнообразные ускорители являются основой высокой производительности многих современных суперкомпьютеров.

По целям применения, современные многоядерные ускорители можно разделить на следующие группы:

• **Специализированные ускорители**. К этому классу относятся сопроцессоры для решения определённых классов задач, например, криптографические ускорители, карты физического моделирования, ИИ-ускорители (для эффективного ускорения решения задач обучения искусственных нейронных сетей, распознавания образов, машинного обучения). Такие ускорители имеют архитектуру, разработанную специально для решения определённых классов задач и малоприменимую для решения других;

• **Графические ускорители**. К этому классу относят многоядерные ускорители, предназначенные для задач обработки компьютерной графики. Несмотря на специфику архитектуры графических сопроцессоров, связанную с особенностями этих задач, их нельзя отнести к специализированным, в связи с развитием в современном мире техник неспециализированных вычислений на графических процессорах (General-purpose computing for graphics processing units, GPGPU);

• **Универсальные ускорители**. Такие ускорители имеют архитектуру, пригодную для решения широкого класса задач, решаемых с использованием центрального процессора, но имеющих потенциал для ускорения за счёт одновременного использования большого количества вычислительных ядер. Примером таких ускорителей может являться семейство сопроцессоров Xeon Phi от Intel.

Ускорители Intel Xeon Phi

Сопроцессоры Intel Xeon Phi являются универсальными ускорителями с архитектурой Intel MIC (Many Integrated Cores). Данная архитектура подразумевает наличие на одном чипе большого количества (> 60) процессорных ядер, поддерживающих набор команд x86, а также дополнительный набор SIMD команд. Ускорители Intel Xeon Phi первого поколения являются платами расширения для слота PCI-Express. Ускорители содержат память GDDR5 объёмом от 8Гб, доступ до которой есть как у процессорных ядер, так и у компьютера-хоста через шину слота PCI-Express [6].

Платы расширения Xeon Phi первого поколения обладают собственной операционной системой, предоставляющей доступ к ускорителю по TCP/IP, что позволяет использовать несколько моделей проведения вычислений с использованием архитектуры Intel MIC. Во-первых, возможна выгрузка данных и исполняемого кода с компьютера-хоста на сопроцессор и выполнение рассчётов на Xeon Phi. Эта модель схожа с программированием графических ускорителей и используется, когда можно обрабатывать большое количество данных с использованием SIMD-команд, доступных в архитектуре. Второй моделью исполнения является непосредственное исполнение. В этом случае, программа собирается для архитектуры Intel MIC специальными инструментами и запускается непосредственно внутри ОС, исполняемой на сопроцессоре. Этот режим используют, если в многопоточном алгоритме, реализуемом программой, есть нетривиальные зависимости между потоками по данным или управлению, однако данный режим не применим для программ, обрабатывающих одновременно большое количество данных, в связи с ограниченным объёмом памяти на плате расширения. Третьей моделью исполнения является "симметричная модель", в рамках которой, компьютер-хост и сопроцессор представляют из себя две вершины в кластере, код выполняется на них одновременно, а обмен данными происходит путём передачи сообщений (например, с использованием MPI).

Графические ускорители с поддержкой технологии CUDA

Программно-аппаратная архитектура параллельных вычислений CUDA была введена корпорацией NVIDIA для использования ее с аппаратными средствами. Компьютерная система CUDA состоит из хоста, которым выступает центральный процессор обычного настольного компьютера и одного или нескольких устройств–графических ускорителей NVIDIA, вычислительных модулей, оснащенных значительным числом арифметико-логических устройств. Функции, исполняемые на графическом ускорителе, называются ядрами (kernels). В отличие от обычных функций языка C, ядра после вызова исполняются N раз параллельно на N исполнительных устройствах. Потоки организованы в пятимерную структуру из двухмерных блоков, состоящих из трехмерных нитей (threads). Блок не может содержать больше 1024 нитей. Синхронизация может быть достигнута только для нитей внутри блока, но не между блоками. Благодаря этому блоки могут выполняться в любом порядке, без необходимости дожидаться друг друга [3].

Вышеописанная схема основана на вычислительной модели SIMT (Single Instruction Multiple Thread), в которой одна инструкция выполняется нитями набора из 32-х нитей (в терминах NVIDIA – «варп»). Если требуется достигнуть максимальной производительности, необходимо следить, чтобы нити внутри варпа не выполняли разные инструкции. Такая ситуация, в частности, возникает, когда в процессе выполнения программы варп натыкается на конструкцию if-else. В таком случае нити начнут выполнять одновременно обе логические ветки if-else. Таким образом, использование CUDA в проекте требует «сильного» распараллеливания кода, а принятие решений о пути исполнения возлагается на хост [4].

Описание эксперимента с использованием ускорителя Intel Xeon Phi

Для решения задачи выполнимости булевых формул с использованием ускорителя Intel Xeon Phi был применён решатель, основанный на решателе с открытым исходным кодом Glucose 4.1, реализующим модификацию алгоритма DPLL[5]. В ходе эксперимента использовалась непосредственная модель выполнения для решателя.

Эксперимент проводился с использованием ускорителя Intel Xeon Phi 7110P со следующими характеристиками:

- 61 процессорное ядро;
- тактовая частота ядер 1,1 ГГц
- 8 Гб память

Таблица 1. Тестовые задачи

№	1	2	3
Число переменных	1000	1040	708
Число дизъюнктов	3600	3668	2664
Выполнимость	SAT	UNSAT	SAT
Источник	Синтезированная	Промышленная	Промышленная
Тип задачи	Случайная 3-КНФ	Анализ электрической цепи на разрыв	Подбор 32-байтного ключа AES

Для каждой тестовой задачи проводилось многократное решение на ускорителе Intel Xeon Phi, с использованием различного числа потоков, с целью исследования масштабируемости решателя при использовании увеличивающегося числа аппаратных ресурсов.

Усреднённые результаты экспериментов приведены на следующих графиках:

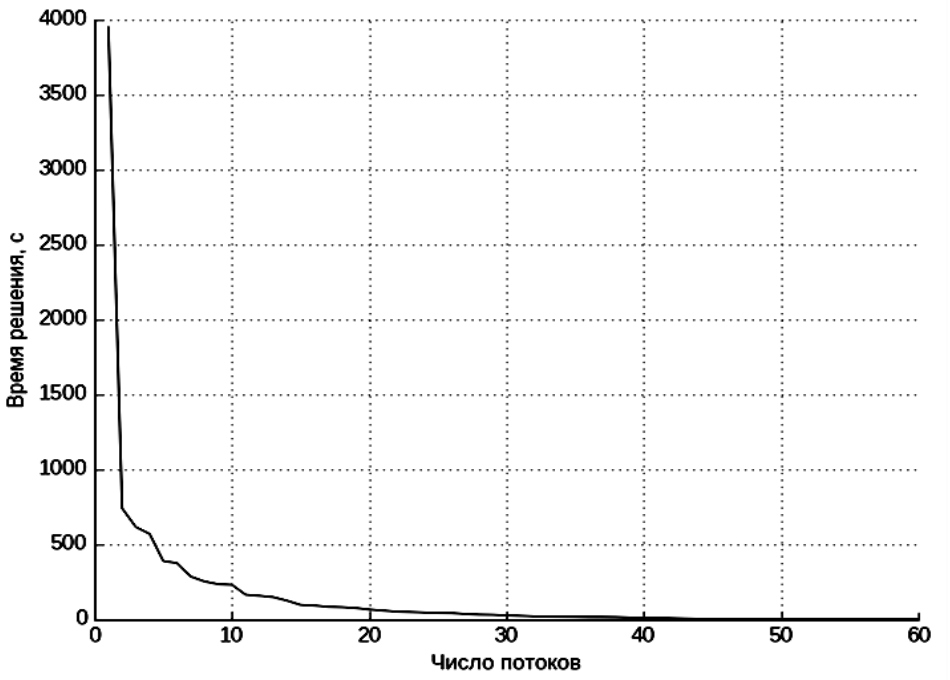

Рис. 1. Решение тестовой задачи 1 с использованием ускорителя Xeon Phi

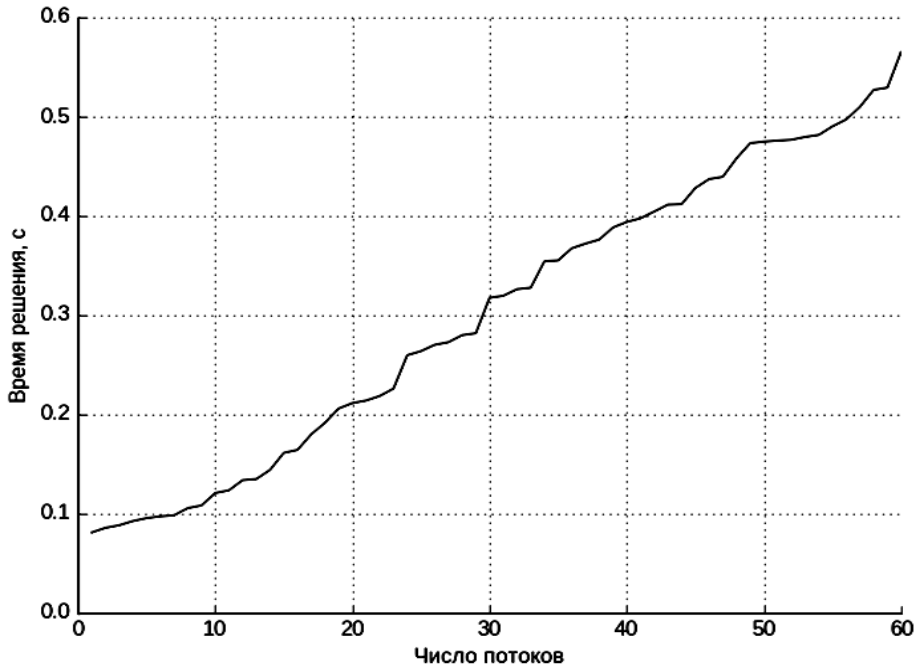

Рис. 2. Решение тестовой задачи 2 с использованием ускорителя Xeon Phi

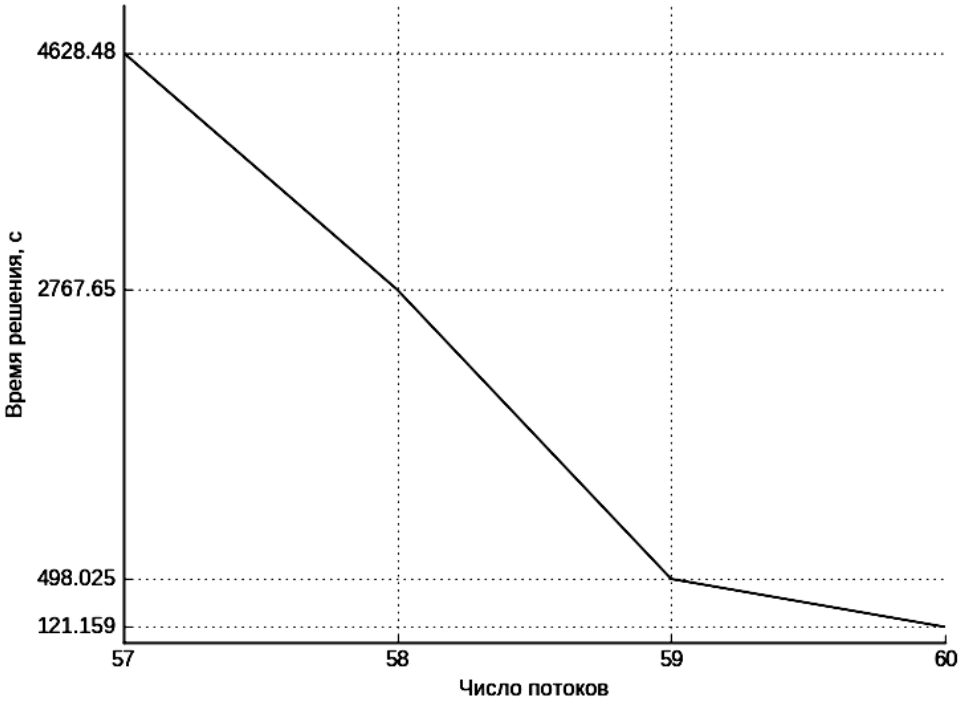

Рис. 3. Решение тестовой задачи 3 с использованием ускорителя Xeon Phi

Из представленных результатов видно, что при использовании ускорителя Intel Xeon Phi для решения крупных задач (задачи 1 и 3), достигается заметное ускорение при увеличении числа потоков в решателе, однако при решении мелких задач (задача 2), использование большего числа потоков замедляет решение.

Описание эксперимента с использованием графического ускорителя

Для решения задачи выполнимости булевых формул с использованием GPU был применен решатель, основанный на алгоритме SLS. Неполный алгоритм был выбран, как более подходящий для архитектуры графических ускорителей.

Эксперимент проводился с использованием компьютера со следующими характеристиками:

- GPU: NVIDIA GeForce GTX 1070
- CPU: Intel Core i5 6600
- 16 Гб ОЗУ

В качестве входных данных использовались случайные формулы с размерностью 75 переменных.

Результаты экспериментов приведены на графике:

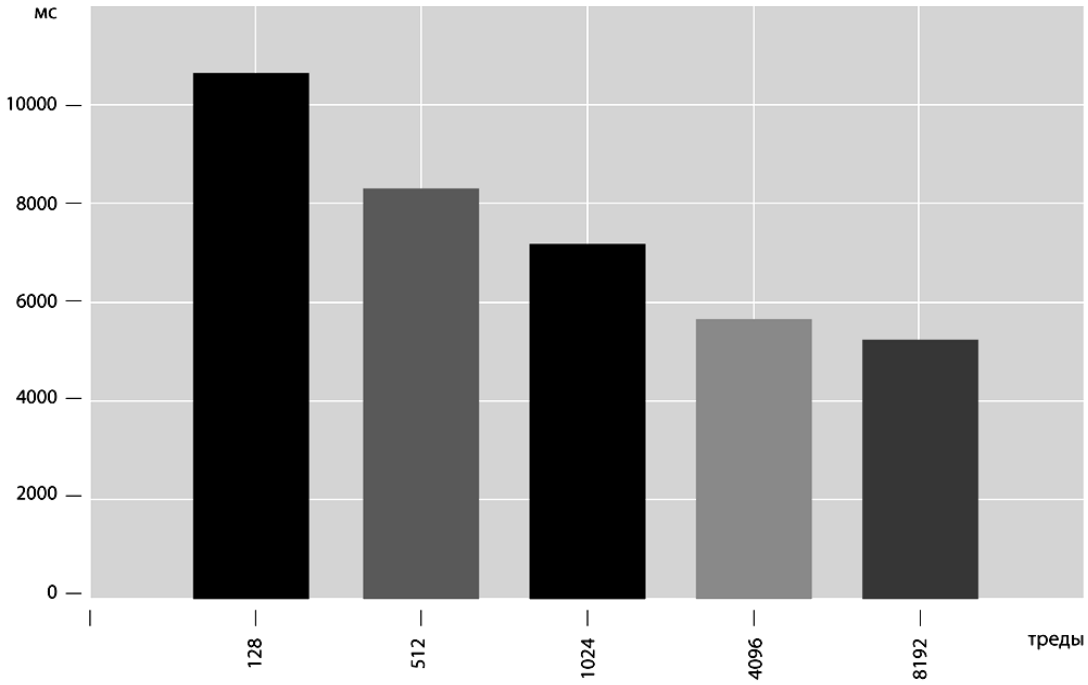

Рис. 4. Ускорение с использованием решателя SAT для платформы NVIDIA CUDA

Как видно из приведенного графика, решатель, работающий на GPU демонстрирует не очень качественную масштабируемость, давая прирост производительности в два раза при увеличении числа потоков в более чем 60 раз.

Выводы

Использование многоядерных ускорителей представляет собой перспективную область исследований проблемы пропозициональной выполнимости. Проведенные эксперименты позволяют рекомендовать к использованию для решения задачи SAT сопроцессорные платы. Использование графических ускорителей менее подходит для данной проблемы, хотя при необходимости это возможно.

Список литературы / References

1. *Davis M., Logemann G., Loveland D.* A Machine Program for Theorem-proving. Commun. ACM 5(7) (July 1962). P. 394-397.
2. *Selman B., Levesque H. and Mitchell D.G.* A new method for solving hard satisfiability problems. In 10th National Conference on Artificial Intelligence, pages 440–446. AAAI Press / The MIT Press. Menlo Park. CA, 1992.
3. CUDA Toolkit Documentation v8.0 — NVIDIA corp., 2017. [Электронный ресурс]. Режим доступа: http://docs.nvidia.com/cuda/ (дата обращения: 03.06.2017).
4. *Costa C.* Parallelization of SAT Algorithms on GPUs. Technical report, Instituto Superior T´ecnico, Lisboa, Portugal (Feb 2013).

5. The Glucose SAT Solver—LaBRI. [Электронный ресурс]. Режим доступа: http://www.labri.fr/perso/lsimon/glucose/ (дата обращения: 03.06.2017).
6. Intel Xeon Phi X100 Family Coprocessor. The Architecture Intel Software. [Электронный ресурс]. Режим доступа: https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner/ (дата обращения: 03.06.2017).
7.