Обзор методов решения задачи планирования параллельных алгоритмов

12, декабрь 2014

Селиверстов Е. Ю.

УДК: 519.6

Россия, МГТУ им. Н.Э. Баумана
theirix@gmail.com

Введение

Важной проблемой, встающей на пути повышения эффективности использования параллельной вычислительной системы, является необходимость распределения вычислительной нагрузки по узлам и коммуникации вычислительных процессов по коммуникационной сети. При построении параллельной программы на основе параллельного метода и алгоритма встают проблемы декомпозиции задачи на подзадачи и планирования подзадач на процессоры целевой системы.

Задачу планирования разделяют на два класса — планирование независимых задач (job scheduling) и планирование взаимодействующих задач (scheduling and mapping) [1]. Планирование независимых задач не требует статического или динамического анализа связей задач и может быть реализовано только динамически. В обзоре рассатриваем задачу планирования взаимодействующих процессов как распределение совокупности взаимосвязанных задач на многопроцессорную или распределенную систему для параллельного выполнения.

1. Модели параллельных программ

Для осуществления задачи планирования и согласования параллельная программа может быть представлена в виде различных моделей [2, 3]. Базовой классической моделью является PRAM (параллельная машина с произвольным доступом) [4]. Для мультипроцессорных систем применяют модели общей памяти, построенные на принципах симметричного мультипроцессирования (SMP) или неоднородного доступа к памяти (NUMA, COMA), модели разделенного глобального адресного пространства (PGAS), модели QSM [5]. Многоядерные и мультиядерные процессоры отличаются сочетанием модели выполнения MIMD, свойственной мультикомпьютерам, и модели общей памяти [6]. Для систем с полностью распределенной памятью распространены модели с неограниченной переда-

чей сообщений, модели взаимодействующих последовательных процессов (CSP). К более реалистичным моделям с передачей сообщений относят модель блочно-синхронного параллелизма (BSP) [7], модель LogP [8], оценивающую загрузку коммуникационной сети. Параллелизм по данным представляется моделями SIMD, потоковой обработки, систолическими массивами и клеточными автоматами. Параллелизм на уровне задач рассматривают в рамках моделей параллелизма задач, основанных на графовом представлении взаимодействующих задач.

Структуру задач и программ в моделях параллелизма задач описывают следующими видами графов [9, 10].

1. Граф зависимостей (task dependency graph). Вершинами графа являются вычислительные задачи (операции), ребрами — зависимости задач.
2. Итерационный граф зависимостей. Конечное множество итераций представлено в виде вершин графа, соединенных ребрами — зависимостями по данным.
3. Граф потока (flow graph). Граф представляет итерации циклического алгоритма. Узлами графа являются операции, ребрами — зависимости по данным внутри итерации и между итерациями.
4. Граф задачи (DAG). Вершинами графа являются задачи, ребрами — коммуникации между задачами. Как правило, вершины и ребра графа задачи имеют веса со смыслом вычислительной сложности и стоимости коммуникации соответственно.
5. Граф взаимодействий задачи (task interaction graph). Граф описывает совокупность процессов и коммуникаций между ними.
6. Графы потока управления (CFG, CDG) [11]. Графы используют при описании потока управления параллельной программы с более сложным, чем линейное, управлением потоком выполнения.
7. Иерархические графы зависимостей [12, 13].
8. Параметризованные графы задачи (PTG) [14, 15]. Модель параметризованного графа представляет из себя символическое представление графа задачи, масштабируемое относительно размеров задач и графов.

Существует возможность преобразования одной графовой модели алгоритма в другую. Основной используемой графовой моделью в задаче планирования является граф задачи. Модель программы на основе графа задачи может быть расширена вытесняющим планированием задач (переназначение запланированной задачи на другие процессоры), учетом дополнительного параллелизма задач в узле, наличием сложного потока выполнения в графе [16].

2. Задача планирования

Процесс планирования возможно рассматривать с различных сторон. Процесс распараллеливания алгоритма [9, 17] обычно состоит из разделения всей задачи на отдельные связанные или несвязанные задачи, анализа зависимостей задач, группировки задач в процессы, отображения задач на вычислительные устройства (mapping) и планирования по-

рядка выполнения (scheduling). Фостер (Foster) [18] в модели РСАМ разделяет процесс на стадии секционирования (partitioning), коммуникации, агломерации и отображения. Саркар (Sarkar) [19] выделяет две фазы — секционирование и планирование. Синнен (Sinnen) [9] рассматривает два принципиальных подхода к решению задачи планирования — списковые эвристики (list scheduling) и кластеризацию (clustering, также называемую агломерацией у Фостера [18]). Алгоритм планирования на основе списковых эвристик состоит из двух фаз: пространственной (выделение процессоров, отображение) и временной (определение времени начала выполнения каждой задач, упорядочивание). Другой класс алгоритмов, основанных на кластеризации, является трехфазным и состоит из кластеризации узлов графа задачи в кластеры, отображения кластеров на процессоры и планирование узлов на процессоре. Синнен отмечает, что разделение эвристик на два класса является условным, и многие алгоритмы могут считаться относящимися сразу к двум классам. Временная фаза в обоих классах алгоритмов обычно реализуется списковыми эвристиками.

Берман в работе [20] рассматривает проблему планирования в общем виде как проблему разрешения отклонения мощности (несоответствия количества процессоров и задач) и отклонения топологии (несоответствия между структурой коммуникаций задач и архитектурой параллельной системы). Алгоритм планирования разрешает сначала отклонение мощности путем редукции графа задач, а затем отклонение топологии путем отображения графов. Ли (Lee) [21] расширяет подход оптимизацией базового отображения с ограничениями, формулируемыми в виде целевых функций.

Сонг Чен (Chen) в работе [22] принципиально разделяет алгоритмы статического планирования на четыре класса в зависимости от того, является ли граф задачи и системы произвольным или специального вида. В наиболее общем случае планирования произвольного графа задачи на произвольную систему возможны два подхода — с внедрением информации о графе системы в граф задачи (машинно-зависимый граф задачи) и без внедрения (более абстрактный машинно-независимый подход, применяемый, например, в [23, 24]). Алгоритмы первого класса рассматриваются, например, в работе Эль-Ревини (El-Rewini) и Али (Ali) [25], сводящих задачу планирования графа задачи с коммуникациями к задаче планирования аугментированного графа задачи без коммуникаций. Ченом в [24] предложен алгоритм второго класса с предварительной кластеризацией как графа задачи, так и графа системы, и последующим отображением с полиномиальной сложностью.

Базовая модель на основе графа задач (также называемая macro-dataflow [19]) неспособна учитывать ограниченные ресурсы коммуникаций, что особенно важно для гетерогенных и многоядерных систем. Усложненные модели на основе графа задач учитывают конфликты коммуникаций, разрешаемые планирование ребер графа, [9] участие процессора в организации коммуникаций [26], конфликты сети [27], конфликты сетевых узлов [28]. Для многоядерных систем дополнительными факторами становятся конфликты кэша [29] конфликты шин [30], конфликты контроллеров памяти и механизма предвыборки [31]. Благодуров в работе [31] отмечает, что достаточно изученные конфликты кэшей оказывают существенно меньшее влияние на планирование, чем прочие указанные факторы.

Проблемы конфликтов ресурсов помимо планировщиков рассматриваются в аналитических моделях производительности. Для многоядерных процессоров Чандра [29] рассматривает эвристические и аналитические вероятностные модели оценки производительности при разделении кэша L2 между многими ядрами. Хи Чен (Xi Chen) [32] предлагает аналитическую вероятностную модель пропускной способности многопоточных мультипроцессоров. Подходы к динамическому планированию потоков для машины с распределенной общей памятью (DSM) на основе графовых моделей сходности потоков изложены Сонгом (Song) и Донгаррой (Dongarra) [33].

Для массивно-параллельных графических процессоров важным фактором производительности оказывается латентность доступа к иерархической памяти [34, 35, 36], конфликты варпов по обращению к памяти [37]. Хонг в работе [35] разрабатывает аналитическую модель параллелизма варпов (MWP) для оценки затрат на передачу данных. Модель Багхсорхи (Baghsorkhi) [38] построена на абстрактном представлении вычислительных ядер в виде расширенных графов потока управления (WFG), получаемых статическим анализом программ. Тесты производительности графического процессора и априорная информация связывают WFG с конкретным процессором. Вычислительные системы на основе нескольких графических процессоров исследуются в работе Дана Шчаа (Schaa) [39]. Предлагается аналитическая модель времени выполнения при варьируемых конфигурациях графических процессоров и задач. Аналитические параметризованные модели используются в адаптивных планировщиках, минимизирующих время выполнения, конфликты памяти или другие факторы. Для идентификации оптимальной модели применяются как однокритериальные стохастические методы оптимизации [40, 36], так и многокритериальные с построением фронта Парето [41].

Джун Занг (Zhang) [42] предлагает другой подход в виде синтеза модели пропускной способности, стоимости инструкций и доступа к памяти на основе тестов производительности. Это позволяет определить возможные узкие места в программе с высокой точности, так как модель базируется на профилировании собственного кода программы.

3. Эвристики планирования задач

Выделяют два основных класса эвристик планирования задач — планирование по списку и кластеризацию [16, 43].

Алгоритмы планирования в основном являются эвристическими, так как задача планирования NP-полная. Бохари [44] отмечает эквивалентность задачи планирования ряду известных NP-сложных проблем (например, изоморфизм графов), решаемых эвристическими методами. Полиномиально-сложными является ограниченный класс задач, например, отображение графов без коммуникаций в виде деревьев на произвольное число процессоров, произвольных графов без коммуникаций на два процессора или интервально упорядоченных графов с коммуникациями [45]. Существуют подходы к решению задачи планирования методами целочисленного программирования (задача отображения) и условной оптимизации (задача планирования) [46, 47], полиномиально сложными методами

оптимизации устойчивого состояния [48], алгоритмами сочетаний пространств характеристик программ и систем [49], комбинированными методами из графо-теоретического представления, линейного программирования и эвристик [47].

В эвристиках планирования по списку задачи сортируют в список в соответствии с приоритетами и зависимостями, а затем итеративно назначают на процессор задачу с выбранным по определенной стратегии приоритетом. Известны две основные эвристики [50] — метод критического пути (выделение в графе задачи критического пути от узла-источника до узла-стока, соответствующего нижней границе времени выполнения задачи) [51, 52] и метод доминирующей последовательности [50]. Динамические эвристики планирования по списку способны улучшать изначальное статическое отображение в процессе выполнения [53]. Для эвристик планирования по спискам в узлах вычисляют так называемые t -уровни (длиннейший путь в графе от узла до узла-истока), b -уровни (длиннейший путь в графе от узла до узла-стока), критические пути.

Смысл эвристик кластеризации заключается в планировании графа задач на идеальную систему с бесконечным числом процессоров. Кластеризация способна выявить локальность вычислений в идеальной системе и минимизировать коммуникации в графе задачи. При отображении кластеров на физические процессоры максимизируется, в свою очередь, параллелизм задачи.

Классическим методом кластеризации является метод минимизации разрезов ребр. К более современным подходам двудольную графовую модель (приспособлена для многофазных разбиений асимметричных зависимостей) [54], гиперграфовые модели (осуществляется минимизация разрезов гиперребр с целью минимизации объема коммуникации в отличие от минимизации числа связей между подмножествами) [55, 56], спектральные методы (разбиение на основании собственных чисел матрицы смежности) [54], асимметричное разбиение (*skewed partitioning*, учитывающее веса в узлах) [57], а также многоуровневые методы [57], основанные на перечисленных и позволяющие проводить разбиение графа последовательной декомпозицией.

Перспективным направлением развития эвристик кластеризаций является применение более адекватных метрик кластеризации (классические модели не учитывают объемы и латентность коммуникаций), многопараметрической кластеризации и параллельных алгоритмов кластеризации [58, 54, 59, 60].

Разнообразные эвристики планирования возможно классифицировать по ослаблению ограничений на граф параллельной системы (классификация предложена Квоком (Kwok) в работе [45]). Так, самым общим классом является APN (произвольная сеть процессоров). Схожими классами эвристик со слабыми ограничениями являются BNP (ограниченное число процессоров, *Bounded Number of Processors*) и UNC (неограниченное число кластеров, *Unbounded Number of Clusters*). Меньшие ограничения накладывают классы алгоритмов дубликации (TDB) и, наконец, алгоритмы для специальных видов графов.

Среди эвристик класса BNP выделяют алгоритм HLFET, эвристику планирования вставки (*Insertion Scheduling Heuristic*), алгоритма модифицированного критического пути

(Modified Critical Path), эвристики «раннее время раньше» (Earliest Time First), алгоритм планирования динамического уровня (Dynamic Level Scheduling), алгоритм локального выделения статических задач (LAST), алгоритм кластеризации на основе декомпозиции графов (CLANS) [61] [16]. Для гетерогенных распределенных систем к эвристикам класса BNP относят также гетерогенный алгоритм «раннее время завершения» (HEFT), алгоритм критического пути (CPOP), алгоритм минимального времени частичного завершения (PCT), алгоритм обобщенного динамического уровня (GDL) [28], алгоритм динамических приоритетов (DPS) [62].

Алгоритмы класса UNC работают с неограниченным числом процессоров, последовательно объединяемых в кластеры для уменьшения длины расписания. После работы алгоритма кластеры отображаются на конечное количество процессоров системы. Таким образом, класс UNC включает как класс эвристик кластеризации [9], так и все другие алгоритмы, ориентированные на неограниченное число процессоров. Известными алгоритмами UNC являются алгоритмы обнуления ребер (Edge Zeroing) [19], кластеризации по одной дуге (Single Edge Clustering) [19], линейной кластеризации (Linear Clustering) [51], алгоритм модифицированного критического пути (MCP) [50], доминирующей последовательности (Dominant Sequence Clustering, DSC) [63], динамического критического пути (Dynamic Critical Path) [52], двухфазный алгоритм Кима (Kim) [64], алгоритмы Cluster-M [65] и Triplet [66] для гетерогенных систем.

Алгоритмы класса TDB (основанные на дублировании задач, Task Duplication Based) отличаются возможностью гарантированно оптимального планировании некоторых структур графа задачи (fork-join).

Задача планирования рассматривает два основных критерия, по которым оптимизируется планирование, — уровень параллелизма (возможность распределения конкурентно выполняющихся задач на большее число процессоров) и объем коммуникаций между процессорами. На общее время выполнения влияет две основные составляющие — время выполнения и время коммуникации. Так, для оценки этих факторов могут применяться различные модели (граф задач, диаграмма Ганта) [67].

Алгоритмы планирования могут учитывать эти критерии как до выполнения графа задач (статическая балансировка), так и непосредственно при выполнении (динамическая балансировка). Статическая балансировка несет меньше накладных расходов, тогда как динамическая может использоваться при переменной вычислительной сложности задач или вычислительной среде с изменяющейся производительностью узлов.

Грама [10] выделяет ряд важных для построения отображения характеристик графа взаимодействия задач (TIG). К характеристикам задач относят их размер (относительная вычислительная сложность, может быть однородной и неоднородной), информацию о размере задач, объем ассоциированных с задачей данных, способ создания (статический или динамический). Взаимодействия между задачами разделяют по времени взаимодействия на статические и динамические, по шаблону взаимодействия на регулярные и нерегулярные. Различные сочетания характеристик графа взаимодействия приводят к решению

задачи либо статического, либо динамического отображения. Выбор между способами отображения также определяется типов декомпозиции данных. Грама выделяет следующие типы.

1. Декомпозиция по данным — разделение входных или выходных данных алгоритма используется для группировки обрабатывающих их задач в кластеры.
2. Рекурсивная декомпозиция — декомпозиция алгоритма на отдельные алгоритмы, решающие подзадачи.
3. Исследующая декомпозиция (exploratory) — разделение области поиска на отдельные подобласти.
4. Спекулятивная декомпозиция — декомпозиция, основанная на вычислении задач со входными данными, соответствующими различным возможным выходным данным предшествующих задач.
5. Гибридные декомпозиции.

4. Планирование для гетерогенных и многоядерных систем

Из интегрированных программных решений для решения задачи планирования в общем виде известны классические пакеты (1980–1990 годы) для мультипроцессорных систем (Hypertool, CASCH, PARSA, OREGAMI, PYRROS, Parallax) [45, 67], современные системы планирования DAG (TrellisDAG [68], Charm++ [69], Hypertool/2 [70], PLASMA [71]). Среди используемых при кластеризации систем разбиения графов отмечают классические (Chaco, METIS), параллельные (ParMETIS, SCOTCH, Jostle, Zoltan, Parkway), гиперграфовые (hMETIS, Zoltan, Parkway) [55, 56, 72, 54, 60].

В последние годы развиваются алгоритмы и методы планирования, разработанные для гетерогенных систем. Гетерогенные системы способны включать разнородные вычислительные устройства: массивно-параллельные процессоры (графические процессоры, программируемые матрицы (FPGA), асимметричные многоядерные процессоры (ASISA)), гетерогенные кластеры и гетерогенные grids. Основными сложностями при планировании на гетерогенных архитектурах являются различные архитектуры и производительность узлов, особенности памяти, более сложный обмен данными между узлами, недетерминированный характер планировщика. Планировщики в гетерогенных системах также учитывают энергопотребление ядер [49].

Задача планирования является ключевой для grid-вычислений. Дополнительными сложностями по сравнению с гетерогенными кластерами является необходимость идентификации ресурсов и учет иерархичности. Алгоритмы планирования, основанные на модели графов задач, используются в подсистемах планирования в пакетах Pegasus, Taverna, GrADS, SimGRID, GridFlow [73], DAGMan [74]. Основные классификации алгоритмов планирования рассматриваются Хамшером (Hamscher) [75], Ю и Байа [76, 73]. Так, алгоритмы возможно разделять по архитектуре планирования (централизованные, децентрализованные, иерархические) [76, 73, 75], по схеме планирования (статические априорные, статические моделирующие, предиктивные динамические и моментально динамические),

по характеру принятия решений о планировании (локальное, уровня узла, или глобальное, уровня всего грида), по стратегии планирования, по способу оценки производительности.

Чен [65] одним из первых рассматривал методологию планирования для гетерогенных суперкомпьютерных систем (парадигма Cluster-M). В работе Беамонта (Beaumont) [28] рассматриваются важные проблемы планирования для распределенных гетерогенных гридов и кластеров. Для статического планирования предлагается применять оптимизацию устойчивого состояния (также [48]) вместо абсолютной минимизации длины расписания, линейную декомпозицию задач на независимые подзадачи и однопортовую (one-port) модель задачи с одновременным вычислением и коммуникацией. Алгоритмы кластеризации (например, Triplet [66]) показывает высокую эффективность по сравнению с списковыми эвристиками.

В работе [77] Грив (Grewe) и О'Бойл (O'Boyle) предлагают статическое планирование для гетерогенной сети из ЦПУ–ГПУ на основе статического анализа исходных OpenCL-программ и предиктивного моделирования иерархии моделей программ машинами опорных векторов (SVM). Система Harmony [78, 79] реализует динамическое планирование задач, представленных в виде вычислительных ядер (kernels), агломерируемых в крупные блоки.

Расширяемая система StarPU для гетерогенных платформ [79] предоставляет модель выполнения, программные средства и каркас для реализации различных стратегий динамического планирования по спискам. Принятие решений при динамическом планировании задач для гетерогенных многоядерных систем использует данные о текущей и исторической загрузке устройств, конкуренции потоков, когерентности кэшей, информации о предыдущих запусках задач, объемах данных. Подходы, основанные на динамическом профилировании, предлагаются в работах Грега (Gregg) [80], Джименез (Jimenez) [81], Бекки (Becchi) [82], Вонга (Weng) [83]. Для гетерогенных систем ASISA известен подход из работы Шепелова (Shepelov) [84] по предварительной разметки задач для быстрого поиска соответствия планировщиком.

Система DAGuE [85]) ориентирована на использование графа задач общего вида и предназначена для полностью распределенного планирования на произвольных гетерогенных системах.

Сзыманек в работе [86] рассматривает гетерогенные алгоритмы для планирования с учетом ограничений по доступу к памяти. В таких алгоритмах используются целевые функции по утилизации фрагментации памяти в отличие от функции минимизации длины расписания для классических алгоритмов.

Для многоядерных и гетерогенных систем широко применяют динамическое планирование на основе графов задач [87, 71, 30]. Основными стратегиями динамического планирования являются перехват задач (task-stealing) [88, 89, 90] и разделение задач (task-sharing) [30], [91]. Стратегии отличаются активной или пассивной ролью процесса по выделению задачи.

В графических процессорах невозможно применить перехват задач из-за особенностей доступа к глобальной памяти, поэтому применяется ряд других подходов. Локальные очереди задач с регулярной легковесной балансировкой предлагаются в работе Лаутербака (Lauterback) [92]. Динамическое формирование варпов в графическом процессоре в работе Фунга (Fung) [37] показывает, что даже при наличии аппаратного планировщика задач, как в графических и многоядерных процессорах, динамические планировщики также имеют возможность улучшать планирование.

Для современных многоядерных и массивно-параллельных графических систем хорошо применимы иерархические планировщики, ориентированные как на иерархию вычислительных устройств [90, 93, 12, 92], так и на иерархию памяти [13, 33]. В работе Саркара [13] предлагается модель иерархических деревьев размещения (НРТ), поддерживающая связь данных и вычислений с различными уровнями иерархии памяти. Хиа [93] рассматривает динамический иерархический планировщик с кластеризатором верхнего уровня и автономными планировщиками уровня групп задач.

Также известны системы параллельного программирования общего вида, ориентированные на динамическое планирование задач и гетерогенные среды: Charm++ [69], Cilk [89], Chapel, X10, Sequoia [94].

Заключение

В обзоре затронуты основные подходы и методы к планированию алгоритмов. Вне рассмотрения остались важные вопросы распараллеливания самих алгоритмов планирования. Задача планирования для кластерных и грид-систем изучена достаточно хорошо. Перспективными направлениями исследований могут быть алгоритмы планирования, в полной мере учитывающие архитектуру массивно-параллельных и гетерогенных мультипроцессорных систем.

Список литературы

1. Kwok Y.-K. Efficient Algorithms for Scheduling and Mapping of Parallel Programs onto Parallel Architectures : Ph. D. thesis / Yu-Kwong Kwok ; The Hong Kong University of Science and Technology. — 1994.
2. Kessler C., rg Keller J. Models for parallel computing: Review and perspectives // Mitteilungen-Gesellschaft für Informatik eV, Parallel-Algorithmen und Rechnerstrukturen. — 2007.
3. Sahni S., Thanvantri V. et al. Parallel computing: Performance metrics and models // Computer & Information Sciences Department University of Florida. — 1995.
4. Harris T. A survey of pram simulation techniques // ACM Computing Surveys (CSUR). — 1994. — Vol. 26, no. 2. — P. 187–206.
5. Gibbons P., Matias Y., Ramachandran V. Can a shared-memory model serve as a bridging model for parallel computation? // Theory of Computing Systems. — 1999. — Vol. 32, no. 3. — P. 327–359.

6. Akhter S., Roberts J. Multi-core programming. — Intel Press, 2006. — Vol. 33.
7. Valiant L. G. A bridging model for parallel computation // Commun. ACM. — 1990. — август. — Vol. 33. — P. 103–111.
8. Parhami B. Introduction to parallel processing: algorithms and architectures. — Springer, 1999. — Vol. 1.
9. Sinnen O. Task scheduling for parallel systems. — Wiley-Interscience, 2007. — Vol. 60.
10. Introduction to Parallel Computing (2nd Edition) / Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta. — 2 edition. — Addison Wesley, 2003.
11. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — БХВ-Петербург, 2002.
12. Vinjamuri S., Prasanna V. Hierarchical dependency graphs: Abstraction and methodology for mapping systolic array designs to multicore processors // Parallel Computing Technologies. — 2009. — P. 284–298.
13. Hierarchical place trees: A portable abstraction for task parallelism and data movement / Y. Yan, J. Zhao, Y. Guo, V. Sarkar // Languages and Compilers for Parallel Computing. — 2010. — P. 172–187.
14. Cosnard M., Loi M. Automatic task graph generation techniques // System Sciences, 1995. Vol. II. Proceedings of the Twenty-Eighth Hawaii International Conference on / IEEE. — Vol. 2. — 1995. — P. 113–122.
15. Cosnard M., Jeannot E. Automatic parallelization techniques based on compact dag extraction and symbolic scheduling // Parallel Processing Letters. — 2001. — Vol. 11, no. 01. — P. 151–168.
16. Khan A., McCreary C., Jones M. A comparison of multiprocessor scheduling heuristics // Parallel Processing, 1994. ICPP 1994. International Conference on / IEEE. — Vol. 2. — 1994. — P. 243–250.
17. Lint B., Agerwala T. Communication issues in the design and analysis of parallel algorithms // Software Engineering, IEEE Transactions on. — 1981. — no. 2. — P. 174–188.
18. Foster I. Designing and building parallel programs. — Addison-Wesley Reading, 1995. — URL: <http://www.mcs.anl.gov/itf/dbpp/>.
19. Sarkar V. Partitioning and scheduling parallel programs for multiprocessors. — MIT press, 1989.
20. Berman F., Snyder L. On mapping parallel algorithms into parallel architectures // Journal of Parallel and Distributed Computing. — 1987. — Vol. 4, no. 5. — P. 439–458.
21. Lee S., Aggarwal J. A mapping strategy for parallel processing // Computers, IEEE Transactions on. — 1987. — Vol. 100, no. 4. — P. 433–442.
22. Chen S., Eshaghian M., Wu Y. Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs // Proceedings of the International Conference on Parallel Processing / Citeseer. — 1995.
23. El-Rewini H., Lewis T. G. Scheduling parallel program tasks onto arbitrary target machines // Journal of parallel and Distributed Computing. — 1990. — Vol. 9, no. 2. — P. 138–153.
24. Chen S., Eshaghian M. M. A fast recursive mapping algorithm // Concurrency: Practice and Experience. — 1995. — Vol. 7, no. 5. — P. 391–409.

25. El-Rewini H., Ali H. H. On considering communication in scheduling task graphs on parallel processors // *Parallel Algorithm And Applications*. — 1994. — Vol. 3, no. 3-4. — P. 177–191.
26. Sinnen O., Sousa L. A., Sandnes F. E. Toward a realistic task scheduling model // *Parallel and Distributed Systems, IEEE Transactions on*. — 2006. — Vol. 17, no. 3. — P. 263–275.
27. Bhat P. B., Raghavendra C. S., Prasanna V. K. Efficient collective communication in distributed heterogeneous systems // *Journal of Parallel and Distributed Computing*. — 2003. — Vol. 63, no. 3. — P. 251–263.
28. Static scheduling strategies for heterogeneous systems / Olivier Beaumont, Arnaud Legrand, Yves Robert et al. // *ISCIS XVII, Seventeenth International Symposium On Computer and Information Sciences*. — 2002. — P. 18–22.
29. Predicting inter-thread cache contention on a chip multi-processor architecture / Dhruba Chandra, Fei Guo, Seongbeom Kim, Yan Solihin // *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on / IEEE*. — 2005. — P. 340–351.
30. Xia Y., Prasanna V. K. Collaborative scheduling of dag structured computations on multicore processors // *Proceedings of the 7th ACM international conference on Computing frontiers / ACM*. — 2010. — P. 63–72.
31. Blagodurov S., Zhuravlev S., Fedorova A. Contention-aware scheduling on multicore systems // *ACM Transactions on Computer Systems (TOCS)*. — 2010. — Vol. 28, no. 4. — P. 8.
32. Chen X. E., Aamodt T. M. A first-order fine-grained multithreaded throughput model // *IEEE International Symposium on High Performance Computer Architecture (HPCA 2009)*. — 2009. — P. 329–340.
33. Song F., Moore S., Dongarra J. Analytical modeling for affinitybased thread scheduling on multi-core platforms // *University of Tennessee, Computer Science Tech. Rep. UT-CS-08-626*. — 2008.
34. A memory model for scientific algorithms on graphics processors / Naga K. Govindaraju, Scott Larsen, Jim Gray, Dinesh Manocha // in *Proc. of the ACM/IEEE Conference on Supercomputing (SC'06)*. — ACM Press, 2006.
35. Hong S., Kim H. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness // *Proceedings of the 36th annual international symposium on Computer architecture*. — ISCA '09. — New York, NY, USA : ACM, 2009. — P. 152–163.
36. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA / Shane Ryoo, Christopher I. Rodrigues, Sara S. Baghsorkhi et al. // *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*. — PPOPP '08. — New York, NY, USA : ACM, 2008. — P. 73–82.
37. Dynamic warp formation and scheduling for efficient gpu control flow / W.W.L. Fung, I. Sham, G. Yuan, T.M. Aamodt // *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture / IEEE Computer Society*. — 2007. — P. 407–420.
38. An adaptive performance modeling tool for gpu architectures / Sara S Baghsorkhi, Matthieu Delahaye, Sanjay J Patel et al. // *ACM Sigplan Notices / ACM*. — Vol. 45. — 2010. — P. 105–114.

39. Schaa D., Kaeli D. Exploring the multiple-gpu design space // *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on / IEEE*. — 2009. — P. 1–12.
40. Choi J. W., Singh A., Vuduc R. W. Model-driven autotuning of sparse matrix-vector multiply on gpus // *ACM Sigplan Notices / ACM*. — Vol. 45. — 2010. — P. 115–126.
41. Program optimization space pruning for a multithreaded gpu / Shane Ryoo, Christopher I. Rodrigues, Sam S. Stone et al. // *Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization*. — CGO '08. — New York, NY, USA : ACM, 2008. — P. 195–204.
42. Zhang Y., Owens J. A quantitative performance analysis model for GPU architectures // *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on / IEEE*. — 2011. — P. 382–393.
43. El-Rewini H., Abd-El-Barr M. *Advanced computer architecture and parallel processing*. — Wiley.com, 2005. — Vol. 42.
44. Bokhari S. On the mapping problem // *Computers, IEEE Transactions on*. — 1981. — Vol. 100, no. 3. — P. 207–214.
45. Kwok Y.-K., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors // *ACM Computing Surveys (CSUR)*. — 1999. — Vol. 31, no. 4. — P. 406–471.
46. Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip / Martino Ruggiero, Alessio Guerri, Davide Bertozzi et al. // *Design, Automation and Test in Europe, 2006. DATE'06. Proceedings / IEEE*. — Vol. 1. — 2006. — P. 6–pp.
47. Chaudhary V., Aggarwal J. A generalized scheme for mapping parallel algorithms // *Parallel and Distributed Systems, IEEE Transactions on*. — 1993. — Vol. 4, no. 3. — P. 328–346.
48. Scheduling strategies for master-slave tasking on heterogeneous processor platforms / Cyril Banino, Olivier Beaumont, Larry Carter et al. // *Parallel and Distributed Systems, IEEE Transactions on*. — 2004. — Vol. 15, no. 4. — P. 319–330.
49. Chen J., John L. K. Efficient program scheduling for heterogeneous multi-core processors // *Proceedings of the 46th Annual Design Automation Conference / ACM*. — 2009. — P. 927–930.
50. Gerasoulis A., Yang T. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors // *Journal of Parallel and Distributed Computing*. — 1992. — Vol. 16, no. 4. — P. 276–291.
51. Kim S., Browne J. A general approach to mapping of parallel computation upon multiprocessor architectures // *International Conference on Parallel Processing*. — Vol. 3. — 1988. — P. 8.
52. Kwok Y.-K., Ahmad I. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors // *Parallel and Distributed Systems, IEEE Transactions on*. — 1996. — Vol. 7, no. 5. — P. 506–521.
53. Maheswaran M., Siegel H. J. A dynamic matching and scheduling algorithm for heterogeneous computing systems // *Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh / IEEE*. — 1998. — P. 57–69.
54. Hendrickson B., Kolda T. Graph partitioning models for parallel computing // *Parallel computing*. — 2000. — Vol. 26, no. 12. — P. 1519–1534.

55. Catalyurek U. V., Aykanat C. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication // *Parallel and Distributed Systems, IEEE Transactions on*. — 1999. — Vol. 10, no. 7. — P. 673–693.
56. Multilevel hypergraph partitioning: applications in vlsi domain / George Karypis, Rajat Aggarwal, Vipin Kumar, Shashi Shekhar // *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*. — 1999. — Vol. 7, no. 1. — P. 69–79.
57. Hendrickson B., Lelandz R., Van Driesschex R. Skewed graph partitioning // *Eighth SIAM conference on parallel processing for scientific computing*. — 1997.
58. Hendrickson B. Graph partitioning and parallel solvers: Has the emperor no clothes? // *Solving Irregularly Structured Problems in Parallel*. — 1998. — P. 218–225.
59. Schloegel K., Karypis G., Kumar V. Parallel static and dynamic multi-constraint graph partitioning // *Concurrency and Computation: Practice and Experience*. — 2002. — Vol. 14, no. 3. — P. 219–240.
60. Zoltan data management services for parallel dynamic applications / Karen Devine, Erik Boman, Robert Heaphy et al. // *Computing in Science & Engineering*. — 2002. — Vol. 4, no. 2. — P. 90–96.
61. Partitioning and scheduling using graph decomposition / C McCreary, J Thompson, H Gill et al. // *Department of Computer Science and Engineering CSE-93-06, Auburn University*. — 1993.
62. Ahmad I., Dhodhi M., Ul-Mustafa R. Dps: Dynamic priority scheduling heuristic for heterogeneous computing systems // *Computers and Digital Techniques, IEE Proceedings- / IET*. — Vol. 145. — 1998. — P. 411–418.
63. Yang T., Gerasoulis A. Dsc: Scheduling parallel tasks on an unbounded number of processors // *Parallel and Distributed Systems, IEEE Transactions on*. — 1994. — Vol. 5, no. 9. — P. 951–967.
64. Kim D., Yi B.-G. A two-pass scheduling algorithm for parallel programs // *Parallel Computing*. — 1994. — Vol. 20, no. 6. — P. 869–885.
65. A selection theory and methodology for heterogeneous supercomputing / Song Chen, Mary Eshaghian, Ashfaq Khokhar, Muhammad Shaaban // *Heterogeneous Processing*. — 1993. — P. 15–22.
66. Cirou B., Jeannot E. Triplet: a clustering scheduling algorithm for heterogeneous systems // *Parallel Processing Workshops, 2001. International Conference on / IEEE*. — 2001. — P. 231–236.
67. El-Rewini H., Ali H., Lewis T. Task scheduling in multiprocessing systems // *Computer*. — 1995. — Vol. 28, no. 12. — P. 27–37.
68. Goldenberg M., Lu P., Schaeffer J. Trellisdag: A system for structured dag scheduling // *Job Scheduling Strategies for Parallel Processing / Springer*. — 2003. — P. 21–43.
69. Kale L. V., Krishnan S. Charm++: Parallel Programming with Message-Driven Objects // *Parallel Programming using C++ / Ed. by Gregory V. Wilson, Paul Lu*. — MIT Press, 1996. — P. 175–213.

70. Wu M.-Y., Shu W., Chen Y. A runtime system for dynamic dag programming // *Parallel and Distributed Processing* / Ed. by Jos'e Rolim. — Springer Berlin Heidelberg, 2000. — Vol. 1800 of *Lecture Notes in Computer Science*. — P. 1192–1199.
71. Song F., YarKhan A., Dongarra J. Dynamic task scheduling for linear algebra algorithms on distributed-memory multicore systems // *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on / IEEE*. — 2009. — P. 1–11.
72. Trifunovic A., Knottenbelt W. J. Parkway 2.0: A parallel multilevel hypergraph partitioning tool // *Computer and Information Sciences-ISCIS 2004*. — Springer, 2004. — P. 789–800.
73. Yu J., Buyya R. A taxonomy of workflow management systems for grid computing // *Journal of Grid Computing*. — 2005. — Vol. 3, no. 3-4. — P. 171–200.
74. Pegasus: Mapping scientific workflows onto the grid / Ewa Deelman, James Blythe, Yolanda Gil et al. // *Grid Computing* / Springer. — 2004. — P. 11–20.
75. Evaluation of job-scheduling strategies for grid computing / Volker Hamscher, Uwe Schwiegelshohn, Achim Streit, Ramin Yahyapour // *Grid Computing—GRID 2000*. — Springer, 2000. — P. 191–202.
76. Yu J., Buyya R. A taxonomy of scientific workflow systems for grid computing // *Sigmod Record*. — 2005. — Vol. 34, no. 3. — P. 44–49.
77. Grewe D., O'Boyle M. A static task partitioning approach for heterogeneous systems using opencl // *Compiler Construction* / Springer. — 2011. — P. 286–305.
78. Diamos G. F., Yalamanchili S. Harmony: an execution model and runtime for heterogeneous many core systems // *Proceedings of the 17th international symposium on High performance distributed computing* / ACM. — 2008. — P. 197–200.
79. Starpu: A unified platform for task scheduling on heterogeneous multicore architectures / C. Augonnet, S. Thibault, R. Namyst, P.A. Wacrenier // *Concurrency and Computation: Practice and Experience*. — 2011. — Vol. 23, no. 2. — P. 187–198.
80. Gregg C., Brantley J., Hazelwood K. Contention-aware scheduling of parallel code for heterogeneous systems // *2nd USENIX Workshop on Hot Topics in Parallelism, HotPar, Berkeley, CA*. — 2010.
81. Predictive runtime code scheduling for heterogeneous architectures / V'ictor J Jim'enez, Llu'is Vilanova, Isaac Gelado et al. // *High Performance Embedded Architectures and Compilers*. — Springer, 2009. — P. 19–33.
82. Becchi M., Crowley P. Dynamic thread assignment on heterogeneous multiprocessor architectures // *Proceedings of the 3rd conference on Computing frontiers* / ACM. — 2006. — P. 29–40.
83. Weng N., Wolf T. Profiling and mapping of parallel workloads on network processors // *Proceedings of the 2005 ACM symposium on Applied computing* / ACM. — 2005. — P. 890– 896.
84. Hass: a scheduler for heterogeneous multicore systems / Daniel Shelepov, Juan Carlos Saez Alcaide, Stacey Jeffery et al. // *ACM SIGOPS Operating Systems Review*. — 2009. — Vol. 43, no. 2. — P. 66–75.

85. Dague: A generic distributed dag engine for high performance computing / George Bosilca, Aurelien Bouteiller, Anthony Danalis et al. // *Parallel Computing*. — 2012. — Vol. 38, no. 1. — P. 37–51.
86. Szymanek R., Kuchcinski K. A constructive algorithm for memory-aware task assignment and scheduling // *Proceedings of the ninth international symposium on Hardware/software codesign / ACM*. — 2001. — P. 147–152.
87. Cab: Cache aware bi-tier task-stealing in multi-socket multi-core architecture / Quan Chen, Zhiyi Huang, Minyi Guo, Jingyu Zhou // *Parallel Processing (ICPP), 2011 International Conference on / IEEE*. — 2011. — P. 722–732.
88. Blumofe R. D., Leiserson C. E. Scheduling multithreaded computations by work stealing // *Journal of the ACM (JACM)*. — 1999. — Vol. 46, no. 5. — P. 720–748.
89. Cilk: An efficient multithreaded runtime system / Robert D Blumofe, Christopher F Joerg, Bradley C Kuszmaul et al. — ACM, 1995. — Vol. 30.
90. Scheduling task parallelism on multi-socket multicore systems / Stephen L Olivier, Allan K Porterfield, Kyle B Wheeler, Jan F Prins // *Proceedings of the 1st International Workshop on Runtime and Operating Systems for Supercomputers / ACM*. — 2011. — P. 49– 56.
91. Squillante M. S., Nelson R. D. Analysis of task migration in shared-memory multiprocessor scheduling. — ACM, 1991. — Vol. 19.
92. Work distribution methods on GPUs : Rep. / Department of Computer Science, University of North Carolina ; Executor: C. Lauterback, Q. Mo, D. Manocha : 2009.
93. Xia Y., Prasanna V., Li J. Hierarchical scheduling of dag structured computations on manycore processors with dynamic thread grouping // *Job Scheduling Strategies for Parallel Processing / Springer*. — 2010. — P. 154–174.
94. Sequoia: programming the memory hierarchy / Kayvon Fatahalian, Daniel Reiter Horn, Timothy J Knight et al. // *Proceedings of the 2006 ACM/IEEE conference on Supercomputing / ACM*. — 2006. — P. 83.