УДК 51.37

Ганкевич И. Г., Дегтярев А. Б.

Метод распределения нагрузки на многопроцессорную систему

Введение. Наиболее разработанным и широко применяемым подходом к распределению нагрузки на вычислительную систему является разбиение данных на однородные части (или разбиение задачи на однородные подзадачи) с последующим распределением их между отдельными ядрами вычислительного устройства или узлами кластера, однако такой подход не всегда работает эффективно. Во-первых, часто общее количество частей, на которые разбиваются входные данные, диктуется не архитектурой и конфигурацией вычислительной системы, а самой задачей и присущими ей ограничениями, и такое распределение не всегда эффективно с точки зрения вычислительной машины: количество частей оказывается либо слишком большим по сравнению с количеством процессоров, работающих параллельно, что ведет к увеличению накладных расходов на обмен данными, либо слишком маленьким, что не позволяет эффективно использовать все доступные вычислительные ядра. Во-вторых, сам характер деления входных данных на части может стать причиной появления неоднородности в размерах различных частей и дисбаланса нагрузки на отдельные вычислительные ядра системы. В-третьих, поскольку в вычислительной системе процессор — не единственное устройство, справляющееся с нагрузкой, и существуют другие компоненты, участвующие в вычислениях (такие как векторные ускорители, видеокарты и устройства хранения), то окончательная производительность системы и время решения конкретной задачи зависят от производительности не только процессоров, но и всех устройств, принимающих участие в вычислениях. Таким образом, учет только лишь процессоров при распределении нагрузки на вычислительную систему является лишь первым приближением к решению задачи о достижении высокой производительности, и учет всех компонент системы позволит улучшить этот результат.

Ганкевич Иван Геннадьевич – аспирант, Санкт-Петербургский государственный университет; e-mail: igankevich@cc.spbu.ru, тел.: +7(921)422-23-21

Дегтярев Александр Борисович – профессор, Санкт-Петербургский государственный университет; e-mail: deg@csa.ru, тел.: +7(911)913-48-99

Алгоритм распределения нагрузки. Для учета производительности всех компонент вычислительной системы и неоднородности различных частей, на которые делятся входные и выходные данные, распределение нагрузки можно провести в два этапа. На первом этапе часть входных данных (или подзадача) направляется на соответствующее устройство, на которое предполагается произвести нагрузку, например, видеокарту или процессор; если же предполагается произвести нагрузку на устройство хранения, то подзадача направляется на процессор или процессорное ядро, специально выделенное под такой тип нагрузки. На втором этапе, когда тип устройства уже выбран, подзадача распределяется на одно из доступных в системе устройств данного типа. Несмотря на то, что на этом этапе подсистема в большинстве случаев является однородной (состоящей из устройств одного типа), для учета неоднородных подзадач необходим алгоритм распределения, который бы учитывал размер частей, на которые делится задача.

Такой алгоритм можно построить на основе алгоритма «заполнения» (англ. backfill), который широко применяется для распределения нагрузки на узлы вычислительного кластера, но для его эффективной работы в случае многопроцессорной системы нужно произвести определенные модификации. Для расчета времени решения задачи на кластере не существует надежного метода, и часто это время задается вручную перед отправкой задачи в очередь [**?**], что неприемлемо в случае многопроцессорной системы, и поэтому время решения отдельных подзадач необходимо предсказать. Для получения надежного предсказания можно воспользоваться любым подходящим статистическим методом и использовать время выполнения предыдущих подзадач в качестве исходных данных. Для учета неоднородной производительности устройств можно воспользоваться тем же методом, только в качестве исходных данных взять производительность устройства на предыдущих задачах (количество задач, завершенных в единицу времени). Чтобы сократить накладные расходы, метод должен быть достаточно простым, и поэтому в тестах был использовано осреднение N последних значений характеристики. Использование статистических методов в случае многопроцессорной системы оправдано, поскольку в отличие от задач, решаемых на кластерах, время решения подзадач достаточно мало, чтобы статистические методы работали эффективно. В случае же кластерных систем ввиду очень большого времени решения одной

задачи использование статистических методов не может дать надежный результат, и алгоритм «заполнения» работает эффективно для небольших задач [?].

Таким образом, распределение нагрузки осуществляется в два этапа: на первом этапе задача направляется на соответствующее ее типу нагрузки устройство, а на втором этапе она направляется на одно из выбранных устройств по алгоритму распределения. Сам же алгоритм является алгоритмом «заполнения» с модификациями, позволяющими автоматически предсказывать время выполнения задачи и производительность устройств.

Нагрузочное тестирование. В качестве задачи для нагрузочного тестирования была выбрана задача генерации взволнованной морской поверхности, имеющая практическое применение в создании виртуального полигона [?, ?, ?, ?]. Генерация взволнованной поверхности реализована как преобразование белого шума, поверхность генерируется с помощью авторегрессионной модели, а давления рассчитываются по аналитической формуле. Алгоритм задачи состоит из трех фаз, наиболее требовательной к вычислительным ресурсам из которых является фаза генерации. Решение начинается с определения коэффициентов авторегрессии и дисперсии белого шума путем решения системы уравнений Юла — Уокера, затем следует генерация белого шума и его преобразование во взволнованную морскую поверхность, которая в завершении записывается в файл. Таким образом, задача заключается в генерации реальной морской поверхности и расчете поля давлений в каждой точке под этой поверхностью и является основой для расчета качки судна и воздействия взволнованной поверхности на стационарный объект.

Программа сбалансированна с точки зрения нагрузки на процессорные ядра, однако, как показали тесты, характеризуется высокой нагрузкой на устройства хранения. До проведения тестирования программа была реализована на OpenMP и для сравнения переписана в соответствии с разработанным подходом к распределению нагрузки, реализованным в виде отдельной библиотеки. Конфигурация оборудования, использованная в тестах, приведена в таблице 1. В результате две реализации были сопоставлены с точки зрения производительности.

Таблица 1. Конфигурация оборудования.

Компонента	Подробности
Язык программирования	C++11
Библиотека потоков	C++11 STL threads
Библиотека атомарных операций	C++11 STL atomic
Подрограммы замера времени	clock_gettime(CLOCK_MONOTONIC) /usr/bin/time -f %e
Компилятор	GCC 4.8.2
Опции компиляции	-std=c++11 -O2 -march=native
Операционная система	Debian 3.2.51-1 x86_64
Файловая система	ext4
Процессор	Intel Core 2 Quad Q9650
Частота процессора (ГГц)	3.00
Количество ядер	4
Емкость ОЗУ (ГБ)	8
Диск	Seagate ST3250318AS
Скорость диска (об./мин.)	7200

В процессе экспериментов была измерена эффективность описанного метода распределения нагрузки, и он показал более высокую производительность в задаче генерации взволнованной поверхности (в задаче генерации большого объема данных) по сравнению с реализацией OpenMP. В результате предыдущих исследований было установлено, что реализация OpenMP имеет наилучшую производительность по сравнению с другими технологиями параллельного программирования [**?**], поэтому эксперимент заключался в сравнении ее производительности с производительностью нового метода на ряде входных данных. При каждом запуске варьировался только размер взволнованной поверхности. В результате эксперимента было установлено, что с увеличением размера поверхности увеличивается разрыв в производительности этих двух реализаций (см. рис. **??**), а высокая производительность предложенного метода обуславливается наложением по времени фазы генерации взволнованной поверхности и фазы записи ее на диск (см. рис. **??**). В реализации OpenMP такого наложения не происходит, и запись на диск начинается сразу после окончания генерации поверхности в отличие от новой реализации, в которой генерация и запись на диск заканчиваются почти одновременно. Таким образом, в программах, работающих с большим объемом данных, конвейеризация параллельных вычислительных фаз более эффективна, чем их последовательное выполнение и позволяет сбалансировать нагрузку не только на процессорные ядра, но и на дисковую подсистему.

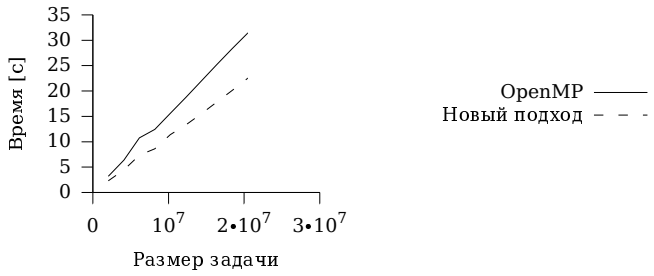

Рис. 1. Сравнение производительности реализаций программы на OpenMP и на разработанной технологии

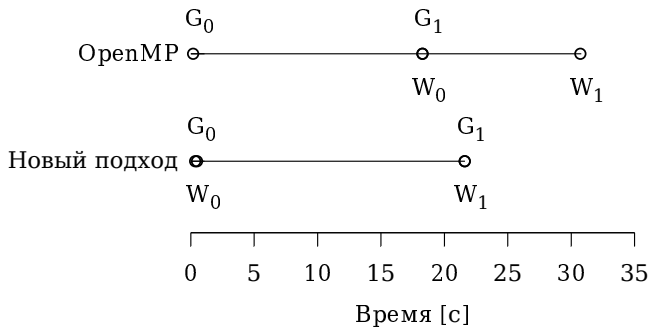

Рис. 2. Наложение параллельных вычислений на $[G_0, G_1]$ и записи данных на диск на $[W_0, W_1]$. В реализации OpenMP наложение отсутствует

Несмотря на то, что технология OpenMP содержит примитивы для создания конвейеров, соединить конвейером две распараллеленные фазы программы можно только вручную. Такое соединение можно реализовать с помощью синхронизированной очереди, в которую направляются сгенерированные части взволнованной поверхности, готовые к записи в файл. Используя директиву *omp section*, можно описать работу каждого из звеньев получившегося конвейера, однако реализовать параллельную обработку каждого из звеньев (или хотя бы первого) не представляется возможным, так как требу-

ется поддержка вложенных директив *omp parallel*, что редко встречается в реализациях OpenMP. Таким образом, реализация описанного метода распределения нагрузки в рамках стандарта OpenMP затруднена.

Выводы. Предложенный метод распределения нагрузки на многопроцессорную систему позволяет получить прирост производительности для приложений, считывающих и записывающих большой объем данных на диск, позволяет сбалансировать нагрузку на процессорные ядра вычислительной системы и назначить различные типы рабочей нагрузки разным процессорным ядрам, а также различным устройствам, в том числе дискам. В дальнейших исследованиях предполагается обобщить этот метод на распределенную вычислительную среду.

Литература

1. Zotkin D., Keleher P. J. Job-length estimation and performance in backfilling schedulers // Proceedings of the 8$^{\text{th}}$ International Symposium on High Performance Distributed Computing. 1999. P. 236–243.

2. Degtyarev A. B., Gankevich I. G. Wave surface generation using OpenCL, OpenMP and MPI // Proceedings of 8$^{\text{th}}$ International Conference «Computer Science & Information Technologies». 2011. P. 248–251.

3. Degtyarev A. B., Reed A. M. Modelling of incident waves near the ship's hull (application of autoregressive approach in problems of simulation of rough seas) // Proceedings of 12$^{\text{th}}$ International Ship Stability Workshop. 2011.

4. Degtyarev A. B., Reed A. M. Synoptic and short-term modeling of ocean waves // Proceedings of 29$^{\text{th}}$ Symposium on Naval Hydrodynamics. 2012.

5. Degtyarev A. B., Gankevich I. G. Evaluation of hydrodynamic pressures for autoregression model of irregular waves // Proceedings of 11$^{\text{th}}$ International Conference «Stability of Ships and Ocean Vehicles». 2012. P. 841–852.