ПРОГРАММНЫЙ КОМПЛЕКС CUBLIC ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, РЕАЛИЗОВАННЫЙ С ПРИМЕНЕНИЕМ ПАРАДИГМЫ ПОТОКА ДАННЫХ

А.Б. Терентьев, С.А. Савихин, С.А. Золотов, В.В. Мохин, А.А. Панкратов, Р.М. Дмитриенко

ООО «Научно-исследовательский центр специальных вычислительных технологий», г. Нижний Новгород

DATAFLOW BASED CUBLIC SOFTWARE PLATFORM FOR HIGH PERFOMANCE COMPUTING SYSTEM

A.B. Terent'ev, S.A. Savikhin, S.A. Zolotov, V.V. Mokhin, A.A. Pankratov, R.M. Dmitrienko

Известны различные методы реализации парадигмы потока данных. Одним из них является метод организации параллельных вычислений, который содержит набор скриптов, данных и правил. Последние управляют перемещением данных между скриптами, которые производят вычисления. Схема перемещения данных сводится к графу, узлами которого служат скрипты [1]. Все промежуточные данные отправляются на жёсткий диск, благодаря чему создаётся прозрачная модель удаления данных, контрольных точек, «горячей» остановки и запуска системы, динамическое управление количеством вычислительных узлов. Недостатком данной методики является регулярное использование жёсткого диска, обмен с которым занимает продолжительное время по сравнению с временем выполнения скриптов и не всегда является необходимым. Стоит отметить, что циклов в явном виде в данном методе организации потока данных нет (они организуются за счёт правил), т. е. циклы переходят в другую сущность.

Другой известной реализацией парадигмы DataFlow является язык потока данных (DFL — Data Flow Language) [2]. Программа на DFL представляет собой набор узлов, каждый из которых включают в себя заголовок и программу обработки данных. Заголовок содержит имя узла, входы для выполнения программы и форму контекста, необходимую и достаточную для выполнения программы. Программа узла производит вычисления, зависящие только от значений входов и контекста. Узлы соединены между собой коммуникационной сетью, по которой перемещаются токены, представляющие собой минимальный объём промежуточных данных со служебной информацией. Данный язык удобен тем, что распараллеливание происходит автоматически в процессе выполнения программы. Кроме того, благодаря организации данных в виде токенов, исчезают циклы на уровне перебора элементов данных. Несмотря на ряд достоинств, метод имеет недостатки, главным из которых является то, что объём «полезной» перемещаемой информации по сравнению с общим объёмом перемещаемых данных значительно мал, что загружает систему и увеличивает время выполнения алгоритма в целом.

В рамках проекта создания инструментальной платформы для разработки и исполнения композитных приложений на гибридных вычислительных системах Cublic была реализована парадигма DataFlow в части управления данными. При этом алгоритм представляется графом, узлами которого являются вычислительные блоки. Данные организованы в виде «полей», которые представляют собой массив векторов, благодаря чему обеспечивается эффективное использова-

ние высоко параллельных структур, в том числе SIMD-архитектура.

Принципиальным отличием предлагаемой реализации DataFlow является оригинальная организация потока данных. Чтобы исключить перемещение по интерконнекту больших объёмов промежуточных данных, используются модификаторы, которые представляют собой набор служебной информации, указывающий каким образом и в каком порядке необходимо изменять (модифицировать) поле данных, чтобы получить новые данные. Таким образом, поле данных хранится без изменений или частично изменённые, а блоки, выполняющие вычисления, формируют новое поле данных путём обратного прохода по модификаторам, часть из которых можно поменять местами. В таком случае перемещается незначительный объём служебной информации (несколько байт, указывающие код и параметры операции) с большим по размеру полем данных (до нескольких мегабайт).

Формально при такой реализации DataFlow алгоритм представляет собой граф. Рёбра графа несут информацию о том, куда и откуда идут данные и, благодаря модификаторам, несут смысловую нагрузку. Выходящие из одного блока (узла) поля, идущие к разным блокам, в конечном итоге могут быть обработаны по-разному.

Если поле содержит большой объём данных, оно будет разделено на несколько частей. Размер каждой части обуславливается с одной стороны оперативной памятью, которой должно хватать для обработки и хранения промежуточных данных. С другой стороны, загружать интерконнект системы малым набором полезных данных с большим объёмом служебной информации не рационально. Кроме того, специальные вычислители (например, графические процессоры) более эффективно обрабатывают большие порции данных.

При такой организации DataFlow циклы как структурный элемент программы исключаются. На уровне перебора элементов массива циклы не требуются благодаря организации данных в виде полей (все элементы вычисляются одновременно). Однако зачастую циклы возникают на более высоком уровне. Для реализации данного рода циклов используется блок Switch, который имеет два входа для получения данных и один выход. Switch можно настроить так, что получив в первый раз данные с одного входа, все остальные данные он будет получать со второго входа. Так организуется цикл в графе. Выходом из цикла может служить блок Fork, который имеет один вход промежуточных данных и два различных выхода. Необходимое количество итераций данные могут пройти по кругу, двигаясь по одному выходу блока, и затем переключиться на другой выход.

Список литературы.

1. Бахтерев М.О., Васев П.А. Методы распределённых вычислений на основе модели потока данных. Прототип системы // XII Международный семинар «Супервычисления и Математическое моделирование»: Тез. докл. Саров, 2010. С. 14 — 16.
2. Климов А.В., Окунев А.С., Левченко Н.Н. Автоматическое распараллеливание линейных циклов путём трансляции в язык потока данных // XII Международный семинар «Супервычисления и Математическое моделирование»: Тез. докл. Саров, 2010. С. 51 — 52.