УДК 004.2; 004.31

А. Ю. Попов

ПРИМЕНЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С МНОГИМИ ПОТОКАМИ КОМАНД И ОДНИМ ПОТОКОМ ДАННЫХ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Приведены принципы организации вычислительных систем с многими потоками команд и одним потоком данных (МКОД), основанные на применении процессора обработки структур данных. Предложена схема взаимодействия устройств системы, обеспечивающая параллельное выполнение потоков команд. На примере алгоритма Дейкстры поиска кратчайших путей рассмотрены особенности разработки программ оптимизации и ход вычислительного процесса в системе МКОД.

E-mail: alexpopov@bmstu.ru

Ключевые слова: *структура данных, оптимизация, алгоритм Дейкстры, вычислительная система с многими потоками команд и одним потоком данных.*

Обработка структур данных является основой многих алгоритмов оптимизации, ускорить работу можно путем повышения степени параллельной обработки [1]. Однако традиционные подходы к проектированию параллельных программ и систем, основанные на применении большого количества однотипных универсальных процессоров, приводят к существенному росту сложности и времени разработки. В настоящее время можно констатировать, что параллельные вычислительные алгоритмы не нашли массового применения.

В работах [2–4] приведены результаты исследований принципов построения и области применения ЭВМ с аппаратной поддержкой операций над структурами данных. Показано, что предложенная архитектура относится к классу систем с многими потоками команд и одним потоком данных. Система, построенная по указанным в работах [2–4] принципам, разработана на основе элементной базы ПЛИС и успешно функционирует. Следующим этапом в процессе внедрения МКОД системы является реализация лингвистического и программного обеспечения, учитывающих особенности данного класса систем. В работе проводится вариант построения и организации вычислений на основе процессора обработки структур данных.

Вычислительная система МКОД состоит из двух процессорных устройств: центрального процессора (ЦП) и процессора обработки структур данных, который называется структурным процессором

(СП). Устройства обмениваются командной информацией и данными через очереди.

Центральный процессор – универсальное процессорное устройство, хранящее в своем оперативно-запоминающем устройстве (ОЗУ) команды и данные. Он выполняет инициализацию системы, запускает задачу на исполнение и обрабатывает результаты алгоритма. Основной вычислительный алгоритм разделяется на два потока команд: команды обработки данных, обрабатываемые ЦП, и команды обработки структур, обрабатываемые СП.

Процессор обработки структур данных подключен к локальным запоминающим устройствам (ЗУ), в которых хранятся структуры данных и команды. ЗУ команд представляет собой адресную память, в которой располагаются отдельные процедуры управления СП, составляющие поток команд обработки структурной информации. Доступ к содержимому адресного ЗУ структур выполняется с помощью уникальных ключей. Отличие СП от ЦП состоит в том, что СП содержит специальные устройства (каталог, операционный буфер и память структур), позволяющие быстро обрабатывать содержимое структур на основе ключей поиска [2]. Хранимая в СП информация представляет собой таблицу «ключ-значение», реализованную на основе Б-деревьев, что обеспечивает высокую скорость основных операций [1]. Для выполнения команд добавления, удаления, поиска и прочих операций с ключом требуется $O(\log n)$ обращений к ОЗУ СП. Следует отметить, что в СП применяются сильно ветвящиеся Б-деревья [1], что позволяет эффективно работать с большими объемами информации. Так, для поиска информации по ключу в структуре из 1,5 млн записей требуется выполнить чтение всего 1*K* байт данных за 16 запросов к ОЗУ на основе DDR/DDR2 SDRAM, что занимает около 1 мкс. Определение мощности множества ключей выполняется без обращений к памяти, а операции объединения, пересечения и дополнения структур требуют $O(n \log n)$ обращений.

В предлагаемом варианте построения вычислительной системы МКОД, показанном на рис. 1, команды СП содержаться в его локальном ЗУ команд и выбираются в естественном порядке следования. Для организации ветвлений набор команд СП включает также команды безусловного перехода, которые поступают из локального ОЗУ или от ЦП. СП не имеет собственного арифметико-логического устройства (АЛУ) и не проверяет условия переходов. Управление ходом вычислений выполняет ЦП, передавая метку перехода в очередь команд управления. Обмен данными между ЦП и СП выполняется через две очереди данных: очередь данных на запись и очередь данных на чтение.

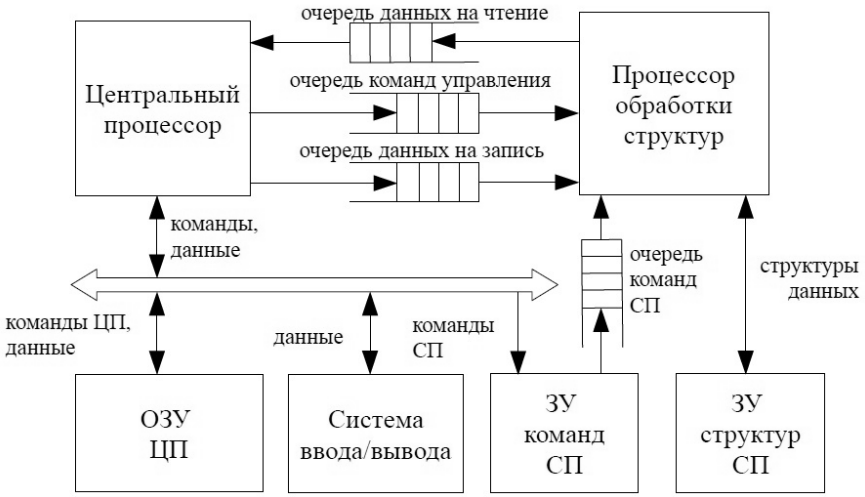


Рис. 1. Вычислительная система МКОД с раздельным хранением команд управления

Очевидно, что такая схема построения МКОД системы обеспечивает параллельную загрузку команд в оба процессора в ходе вычислительного процесса, а значит позволяет достичь максимальной параллельности при выполнении потоков. Однако для этой системы следует разделить последовательную программу на две взаимосвязанные программы, что требует применения специализированных средств компиляции.

Для пояснения принципов работы системы рассмотрим реализацию алгоритма Дейкстры для поиска кратчайших путей на графе. Этот алгоритм используется, в частности, для сетевой маршрутизации и поиска кратчайших маршрутов в навигационных системах. Задача формулируется следующим образом. Для взвешенного ориентированного графа $G(V, E)$ без петель и дуг отрицательного веса найти кратчайшие пути от некоторой вершины до всех остальных. Пусть: $w[u][v]$ – вес ребра, соединяющего вершину u и v; $Adj[u]$ – множество вершин, смежных с u; s – исходная вершина; Q – множество вершин, которые осталось рассмотреть для поиска кратчайших путей; $d[u]$ – расстояние от вершины s до вершины u; $p[u]$ – вершина, предшествующая вершине u в кратчайшем пути от s до u.

Рассмотрим работу алгоритма для систем с одним потоком команд и одним потоком данных (ОКОД). В начальный момент времени множество Q состоит из всех вершин графа: $Q = V$. Алгоритм предполагает на каждом шаге поиск в множестве Q вершины u с наименьшим значением $d[u]$, ее удалению из Q, а также вычислению значений $d[v]$ для всех связанных с u вершин, входящих в Q. Если полученная длина пути короче ранее найденной, то она модифицируется и сохраняется новый маршрут $p[v]$ через вершину u. В итоге, ко-

гда будут просмотрены все вершины и Q останется пустым, в $p[u]$ окажется кратчайший маршрут, а в $d[u]$ – его длина. Псевдокод алгоритма представлен ниже:

```
АЛГОРИТМ ДЕЙКСТРЫ ОКОД(G,s,Q,d,p)
Q=V; d[s]=0; p[s]=0;
ЦИКЛ (для всех вершин u∈V & u≠s)
    d[u]=∞;
ВСЕ ЦИКЛ
ЦИКЛ ПОКА (Q≠∅)
    ПОИСК (u∈Q с минимальным значением d[u])
    Q=Q\u;
    ЦИКЛ (для всех вершин v, v∈Adj[u] & v∈Q)
            ЕСЛИ (d[v]>d[u]+w[u][v]) ТО
            d[v]=d[u]+w[u][v];
            p[v]=u;
        ВСЕ ЕСЛИ
    ВСЕ ЦИКЛ
ВСЕ ЦИКЛ
КОНЕЦ
```

В алгоритме необходимо реализовать структуры данных для хранения графа G, множеств Q, множеств длин путей $w[u]$ из вершин к смежным вершинам, информацию о кратчайших путях $p[u]$ и их длинах $d[u]$ для каждой вершины. Как говорилось ранее, для вычислительной системы МКОД требуется представить их в виде таблиц «ключ-значение».

Примем следующие обозначения: S.KEY(k_1, .., k_n) – составной ключ поиска в структуре S, состоящий из полей k_1, .., k_n; S.DATA(d_1, ..., d_n) – данные для структуры S, состоящие из полей d_1, ..., d_n. Доступ к конкретным полям данных может быть продемонстрирован в псевдокоде алгоритма с помощью модификатора, т. е. как S.DATA.d_1. Например, изменение поля x в составном поле данных указывается как S.DATA.x=10.

Множество Q используется для поиска вершины с наименьшим показателем $d[u]$. Поскольку значение $d[u]$ для разных вершин в структуре Q может быть одинаковым, то $d[u]$ не должен являться ключом. В качестве ключа может быть выбрана пара значений Q.KEY=($d[u]$,u). Это обеспечивает не только поиск вершины с минимальным $d[u]$, но и выбор вершины с наименьшим номером из нескольких возможных вариантов. Поле данных структуры Q не используется: Q.DATA=(0).

Исходный граф G в алгоритме служит для определения списка смежных вершин и другой информации, соответствующей каждой

вершине. Ключом поиска в структуре G является уникальный номер вершины в графе G: G.KEY=(u), а данными для структуры G – множество смежных вершин $Adj[u]$, массив длин путей для них $w[u]$, найденное кратчайшее расстояние $d[u]$, маршрут $p[u]$, а также бит принадлежности вершины к множеству Q: т.е. G.DATA=($Adj[u]$, $w[u]$, $d[u]$, $p[u]$, $u \in Q$). Пояснить использование этих структур можно на примере поиска кратчайших путей для графа, представленного на рис. 2.

Структура G

Ключ	Значение				
u	Adj[u]	w[u]	d[u]	p[u]	u∈Q
1	2,3	2,7	0	0	1
2	1,3,4	2,4,1	∞	0	1
3	1,2,4,5	7,4,2,7	∞	0	1
4	2,3,5	1,2,6	∞	0	1
5	3,4	7,6	∞	0	1

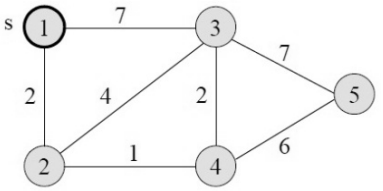
а

Структура Q

Ключ	
d[u]	u
0	1
∞	2
∞	3
∞	4
∞	5

б

Граф



в

Рис. 2. Пример поиска кратчайших путей в графе (начальное состояние структур данных):

а – структура G; *б* – структура Q; *в* – граф

На первом шаге алгоритма из структуры Q выбирается минимальный ключ Q.KEY=(0,1) и по нему определяется код u, соответствующий вершине s = 1. Для этой вершины в структуре G выбирается строка и определяются поля $Adj[u]$,$w[u]$, $d[u]$,$p[u]$,$u \in Q$. Найденная строка исключается из структуры Q по известному ключу Q.KEY=($d[u]$,u). Далее, для каждой вершины v из множества $Adj[u]$ по структуре G определяется принадлежность к множеству Q (параметр $u \in Q$). Если $u \in Q$ – истино, то проверяется условие $d[v]$ = = $d[u] + w[u][v]$, где $w[u][v]$ – один из элементов множества $w[u]$, соответствующий ребру между u и v. Результат работы алгоритма после первой итерации показан на рис. 3. Стрелками на графе указаны отрезки найденного маршрута.

Структура G

Ключ	Значение				
u	Adj[u]	w[u]	d[u]	p[u]	u∈Q
1	2,3	2,7	0	0	0
2	1,3,4	2,4,1	2	1	1
3	1,2,4,5	7,4,2,7	7	1	1
4	2,3,5	1,2,6	∞	0	1
5	3,4	7,6	∞	0	1

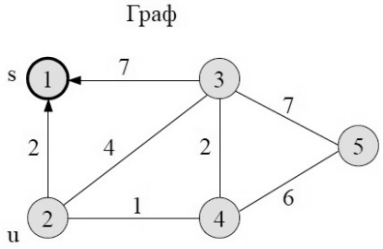
а

Структура Q

Ключ	
d[u]	u
2	2
7	3
∞	4
∞	5

б

Граф



в

Рис. 3. Пример поиска кратчайших путей в графе (состояние структур данных после выполнения первой итерации)

а – структура G; *б* – структура Q; *в* – граф

Цикл повторяется до полного опустошения структуры Q. В результате будут определены кратчайшие пути и их длины для всех вершин. Состояние структур после выполнения всех итераций представлено на рис. 4.

Структура G

Ключ	Значение				
u	Adj[u]	w[u]	d[u]	p[u]	u∈Q
1	2,3	2,7	0	0	0
2	1,3,4	2,4,1	2	1	1
3	1,2,4,5	7,4,2,7	4	5	1
4	2,3,5	1,2,6	2	3	1
5	3,4	7,6	9	4	1

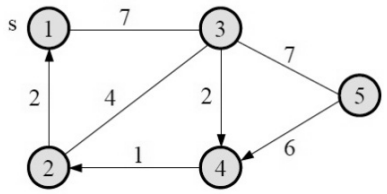
а

Структура Q

Ключ	
d[u]	u

б

Граф



в

Рис. 4. Пример поиска кратчайших путей в графе (состояние структур в конце работы алгоритма)

а – структура G; *б* – структура Q; *в* – граф

Используя указанные структуры можно преобразовать последовательный вариант алгоритма Дейкстры в два параллельных потока команд, исполняемые ЦП и СП. В приведенном ниже псевдокоде приняты обозначения, поясняющие ход вычислительного процесса. Операции ПОЛУЧИТЬ и ПЕРЕДАТЬ используются для обращения к очередям данных и команд. Как показано ранее, обмен информацией между двумя процессорами происходит посредством очередей, которые обеспечивают синхронизацию исполнения этих ветвей. Например, если в алгоритме присутствует команда ПОЛУЧИТЬ, то устройство ожидает поступление данных, не выполняя других действий. Идеология организации вычислений на основе передачи сообщений через очереди позволяет использовать для описания таких процедур синтаксис библиотеки MPI (Massage passing Interface), принятой в качестве основы взаимодействия параллельных ветвей алгоритмов в современных вычислительных системах. В представленном параллельном варианте детализированы фазы инициализации и сохранения результата.

```
АЛГОРИТМ ДЕЙКСТРЫ МКОД ЦП(G,s,Q,d,p)
//Инициализация
ЦИКЛ ПОКА (для всех вершин u∈V & u≠s)
    ПЕРЕДАТЬ (Q.KEY=(∞,u), Q.DATA=(0))
    ПЕРЕДАТЬ (G.KEY=(u),G.DATA=(Adj[u],w[u],∞,0,1))
ВСЕ ЦИКЛ
ПЕРЕДАТЬ (Q.KEY=(0,s), Q.DATA=(0))
ПЕРЕДАТЬ (G.KEY=(s), G.DATA=(Adj[s],w[s],0,0,1))
```

```
//Завершить инициализацию
ПЕРЕДАТЬ (МЕТКА1)
//Основной цикл
ПОЛУЧИТЬ |Q|
ЦИКЛ ПОКА (|Q|>0)
ПОЛУЧИТЬ(d[u],u)
    ПЕРЕДАТЬ (G.KEY=(u))
    ПОЛУЧИТЬ(Adj[u],w[u],d[u],p[u],u∈Q)
    ЦИКЛ (для всех вершин v, v∈Adj[u])
        ПЕРЕДАТЬ (G.KEY=(v))
        ПОЛУЧИТЬ (Adj[v],w[v], d[v], p[v],v∈Q)
        ЕСЛИ (v∈Q & d[v]>d[u]+w[u][v]) ТО
            ПЕРЕДАТЬ (G.KEY=v, G.DATA=(Adj[v],
            w[v],d[u]+w[u][v],u,0))
            ПЕРЕДАТЬ (Q.KEY=(d[u]+w[u][v],v))
        ИНАЧЕ
        //Переход к следующей итерации
            ПЕРЕДАТЬ (МЕТКА2)
        ВСЕ ЕСЛИ
    ВСЕ ЦИКЛ
    //Переход к следующей итерации
    ПЕРЕДАТЬ (МЕТКА3)
    ПОЛУЧИТЬ |Q|
ВСЕ ЦИКЛ
//Завершение работы
ПЕРЕДАТЬ МЕТКА4
КОНЕЦ
```

Центральный процессор передает в СП ключи поиска и данные для хранения и получает в ответ результаты выполнения команд. Сами команды управления обработкой структур данных, как было сказано ранее, сохранены в ЗУ структур СП. Поскольку в алгоритме предполагается выполнить циклическую обработку и ветвление в зависимости от результатов обработки потока данных, ЦП выполняет соответствующие проверки. Например, ЦП проверяет условия выхода из циклов и условия ЕСЛИ ($v \in Q$ & $d[v] > d[u] + w[u][v]$) ТО … ИНАЧЕ .... После проверки условий ветвления ЦП либо продолжает текущую ветвь, что не меняет порядок обработки команд управления СП, либо передает сообщение о принудительном переходе на обработку следующей итерации цикла. Это сообщение должно быть соответствующим образом обработано в СП. Вторая ветвь алгоритма, исполняемая на СП, показана ниже:

```
АЛГОРИТМ ДЕЙКСТРЫ МКОД СП(G,s,Q,d,p)
//Инициализация
ЦИКЛ
    ПОЛУЧИТЬ((Q.KEY,Q.DATA))
    ДОБАВИТЬ(Q.KEY,Q.DATA)
    ПОЛУЧИТЬ (G.KEY,G.DATA)
    ДОБАВИТЬ(G.KEY,G.DATA)
ВСЕ ЦИКЛ
//Основной цикл
МЕТКА1:
ЦИКЛ
    ПЕРЕДАТЬ |Q|
    ПОИСК МИНИМУМ (Q)
    ПЕРЕДАТЬ МИНИМУМ (Q)
    УДАЛИТЬ МИНИМУМ (Q)
    ПОЛУЧИТЬ G.KEY
    ПОИСК (G.KEY, G.DATA)
    ПЕРЕДАТЬ (G.KEY, G.DATA)
    ЦИКЛ
        ПОЛУЧИТЬ(G.KEY)
        ПОИСК (G.KEY, G.DATA)
        ПЕРЕДАТЬ (G.KEY, G.DATA)
        ПОЛУЧИТЬ(G.KEY,G.DATA)
        ИЗМЕНИТЬ(G.KEY,G.DATA)
        ПОЛУЧИТЬ(Q.KEY,Q.DATA)
        УДАЛИТЬ(Q.KEY,Q.DATA)
        ПОЛУЧИТЬ(Q.KEY,Q.DATA)
        ДОБАВИТЬ(Q.KEY,Q.DATA)
МЕТКА2:
    ВСЕ ЦИКЛ
МЕТКА3:
ВСЕ ЦИКЛ
МЕТКА4: КОНЕЦ
```

Как видно из приведенного псевдокода, многие участки кода СП не требуют синхронизации с ЦП, а обрабатываются независимо. Эта особенность хорошо прослеживается на последовательных участках команд чтения в основном цикле программы. Такие команды могут запускаться на исполнение в асинхронном режиме без получения разрешающего сигнала на запуск от ЦП. Команды могут выполняться до тех пор, пока выходная очередь результатов (очередь данных на чтение на рис. 1) не будет заполнена. В случае выхода из цикла по сигналу от ЦП (например, переход к МЕТКА 2) очередь принудительно освобождается для подготовки к принятию новых данных.

Команды управления СП, предполагающие запись данных, выполняют также функцию синхронизации, т. к. данные должны поступить из ЦП.

Таким образом, из проведенного анализа варианта реализации алгоритма Дейкстры для МКОД систем очевидно, что возможно параллельное выполнение алгоритмов оптимизации, основанное на исполнении двух потоков команд.

Анализ полученного псевдокода показывает, что преобразование последовательной программы для ОКОД систем в две параллельные программы может быть формализовано, а значит и автоматизировано. В связи с этим целесообразно проведение дальнейших исследований в области разработки методологии построения средств параллельного программирования для МКОД систем.

СПИСОК ЛИТЕРАТУРЫ

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2000. – 960 с.
2. Попов А. Ю. Реализация электронной вычислительной машины с аппаратной поддержкой операций над структурами данных // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. Спец. вып. «Информационные технологии и компьютерные системы» – 2011. – С. 83–87.
3. Попов А. Ю. Электронная вычислительная машина с аппаратной поддержкой операций над структурами данных // Аэрокосмические технологии: Научн. материалы Второй международной научно-технической конференции, посвященной 95-летию со дня рождения академика В.Н. Челомея – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – С. 164–165.
4. Электронная вычислительная машина с многими потоками команд и одним потоком данных / Попов А.Ю. Пат. 71016 Рос. Федерация. №2006115810. Заявл. 10.05.2006; Опубл. 20.02.2008. Бюл. № 5. 1 с.

Статья поступила в редакцию 14.05.2012