УДК 004.382.2

Е.Г. Бryндин

Исследовательский центр «Естествоинформатика», Новосибирск, Россия
Национальная Суперкомпьютерная Технологическая Платформа

РЕШЕНИЕ ПРОБЛЕМЫ НЕПРЕРЫВНОЙ ОБРАБОТКОЙ ПРОГРАММ НА ВИРТУАЛЬНОЙ ПАМЯТИ

В статье рассматривается решение проблемы управления непрерывной автоматизированной обработкой больших объемов информации на виртуальной памяти без задержки получения результатов из-за обменов. Непрерывная обработка больших объемов информации на виртуальной памяти считалась технически нереализуемой.

Цель данного исследования показать, что проблема не техническая, а алгоритмическая. Задачи исследования: показать универсальное решение проблемы, определить архитектуру целевой оптоэлектронной супер-ЭВМ, определить структуру рабочих программ.

Ключевые слова: упреждающее замещение модулей на виртуальной памяти, непрерывная обработка программ, суперкомпьютер с упреждающим управлением памятью.

E.G. Bryndin

Research center "Estestvoinformatika", Novosibirsk, Russian Federation
National Supercomputer Technological Platform

SOLUTION CONTINUOUS PROCESSING PROGRAMS ON VIRTUAL MEMORY

In article the solution of the problem of management of the continuous automated processing of large volumes of information on virtual memory without delay of receiving results because of exchanges is considered. Continuous processing of large volumes of information on virtual memory was considered as the technically not realized.

Objective of this research to show that a problem not technical, but algorithmic. Research problems: to show a universal solution, to define architecture of the target optoelectronic supercomputer, to define structure of working programs.

Keywords: anticipatory replacement of modules on virtual memory, continuous processing of programs, a supercomputer with anticipatory management of memory.

1. Непрерывная обработка на виртуальной памяти программ с детерминированно-связанными модулями. Рассмотрим вычислительные процессы с упреждающим управлением виртуальной памятью. Вычислительные процессы с упреждающим управлением виртуальной памятью обрабатывают программы с детерминированно-связанными модулями [1, 2].

Упреждающий метод управления памятью использует:

– детерминированный, динамический метод распределения ресурсов оперативной памяти;

– упреждающие акты анализа связности модулей программы;

– упреждающий запрос, упреждающий метод параллельного замещения обработанных модулей программы;

– потоковое упреждающее перемещение значений общих данных между модулями на оперативных сегментах памяти.

Программы с детерминированно-связанными модулями допускают упреждающие анализ цепочек связей, поиск модулей, перемещение модулей и значений общих данных глубины m по текущим данным, где m – число сегментов оперативной памяти. Упреждающий метод управления позволяет обрабатывать непрерывно программу с детерминированно-связанными модулями на виртуальной памяти. Модули программы обрабатываются последовательно, обработка каждого модуля параллельная.

Критерий непрерывной обработки. Вычислительные процессы с упреждающим методом управления памятью непрерывно обрабатывают программу с детерминированно-связанными модулями, если время обработки каждой последовательности из m связных модулей больше времени анализа связей, перемещения общих данных и параллельного замещения m модулей на 2 m сегментах оперативной памяти.

Метод замещения модулей программ с детерминированно-связанными модулями на оперативной памяти, упреждающий их обработку, позволяет обеспечить непрерывность обработки на виртуальной памяти по критерию непрерывности, в отличие от всех используемых методов замещения модулей программ со случайными обращениями к модулям. Вычислительные процессы с упреждающим методом управления памятью – это новое направление в информатике автоматизированной обработки больших программ с детерминированно-связанными модулями.

2. Программы с детерминированно-связанными модулями. Пусть задана модульно-связанная программа P, где $P = \{PI_j\}$, $j = 1,\ldots,n$, PI_j – либо операционный модуль, либо модуль данных программы, либо модуль ввода-вывода. $PI_j = \{UI_j, OD_j, LD_j, P_j, SPP_j, SPI_j\}$, где UI_j – управляющая информация, OD_j – общие данные, LD_j – локальные данные, P_j – программа обработки, SPP_j – связь по поиску модуля следующего по исполнению, SPI_j – алгоритмическая связь модулей по передаче исполнения.

Определение 1. Последовательность модулей $PI1 \ldots PIj \ldots PIk$, связанных через программы поиска $SPP1 \ldots SPPj \ldots SPPk$, назовем модульным исполнением PPI программы P, если последовательность программ обработки $P1 \ldots Pj \ldots Pk$ определяет результат программы P по ее данным, где $l = 1 \ldots m$.

Определение 2. Структура связей по поиску модулей программы P детерминирована, если по ее данным можно выполнить последовательности $\{(SPP_1 \ldots SPPj \ldots SPP_k)^l\}$ программ поиска модулей для всех модульных исполнений $\{PPI^l\}$ программы P.

Определение 3. Модульно-связанная программа P с определенным размером всех модулей называется программой с детерминированно-связанными модулями, если структура связей по поиску ее модулей детерминирована.

2.1. Операторные схемы для создания программ с детерминированно-связанными модулями. Разработана теория операторных схем с естественной интерпретацией для преобразования любой программы в программу с детерминированно-связанными модулями.

Определение 4. Ориентированный граф $G(X,\Gamma,U)$, в котором операторы программы P представлены вершинами $X = \{O_1,O_2,\ldots,O_n\}$ графа, а информационные связи – по данным заданы функцией Γ: $X \to Y$, а функцией U: $X \to Y$ – связи по исполнению между операторами заданы, *назовем операторной схемой программы P*.

Определение 5. *Оператор O_j альтернативно управляется по исполнению*, если он имеет несколько входов по исполнению.

Определение 6. *Оператор O_j альтернативно управляет исполнением*, если он имеет несколько выходов по исполнению.

Определение 7. *Оператор O_j называется полисемантическим*, если он обладает различными семантиками исполнения.

Определение 8. Оператор Oj называется полиинформационным, если он имеет несколько информационных связей по передаче результата в качестве операндов (аргументов) другим операторам.

2.2. Типы операторных схем

Определение 9. Операторная схема $G(X,\Gamma,U)$ называется *линейной* по исполнению, если все операторы в схеме однозначно управляются по исполнению и однозначно управляют исполнением.

Определение 10. Операторную схему $G(X,\Gamma,U)$, состоящую из независимых линейных операторных подсхем, *назовем линейно-образной* по исполнению операторной схемой.

Определение 11. Операторная подсхема $G(X_i,\Gamma,\Gamma)$, имеющая один оператор входа через связи по исполнению, альтернативно управляющий двумя или более операторами подсхемы, и один оператор выхода из подсхемы через связи по исполнению, альтернативно управляемый двумя и более операторами подсхемы, причем операторы входа и выхода различные, называется *гамакообразной* по исполнению операторной подсхемой.

Определение 12. Операторная подсхема $G((X0 \subset X), \Gamma,U)$, у которой операторы $X0 =\{Oj\}$ образуют одну замкнутую цепочку ($Oj1$, $Oj2,\ldots,Ojn$) через связи по исполнению $U{:}X0{\rightarrow}X0$, называется *несцепленной операторной подсхемой с возвратом по исполнению*.

Определение 13. Операторные подсхемы $G((X_1 \subset X), \Gamma,U)$ и $G((X_2 \subset X),\Gamma,U)$ с возвратом по исполнению называются *сцепленными*, если подсхемы $X_1 \cap X_2 = \varnothing$ имеют общие операторы.

Определение 14. Нелинейно образную операторную схему $G(X,\Gamma,U)$ без гамакообразных подсхем по исполнению и без подсхем с возвратом по исполнению назовем *линейно перекрестной операторной схемой по исполнению*.

Определение 15. Исполнение $Oi1,Oi2,\ldots,Oin$ операторной схемы $G1(X1,\Gamma1,U1)$ и исполнение $Oj1,Oj2,\ldots,Ojn$ операторной схемы $G2(X2,\Gamma2,U2)$ называются *равносильными*, если для одинаковых входных данных приводят к одинаковым результатам.

Определение 16. Операторные схемы $G1(X1,\Gamma1,U1)$ и $G2(X2,\Gamma2,U2)$ называются *равносильными*, если множества их исполнений равносильные.

Определение 17. Программы $P1$ и $P2$, имеющие равносильные операторные схемы, называются *равносильными*.

Определение 18. Преобразование операторной схемы программы $P1$ в операторную схему равносильной программы $P2$ называется преобразованием операторных схем по алгоритмическому базису.

Определение 19. Преобразование операторной схемы $G(X,\Gamma,U)$, при котором из подсхемы $G(X1,\Gamma,U)$ операторной схемы $G(X,\Gamma,U)$ получается полисемантический полиинформационный оператор с сохранением всех информационных связей и связей по передаче исполнения с операторной схемой $G(X,\Gamma,U)$, назовем синтезом.

Определение 20. Преобразование операторной подсхемы в несколько независимых подсхем с сохранением всех информационных связей и связей по передаче управления с операторной схемой назовем расщеплением.

Определение 21. Подсхема операторной схемы, обрабатывающая группу входных данных независимо от дополнительной подсхемы, называется автономной по входным данным.

Теорема 1. В операторной схеме $G(X,\Gamma,U)$ можно выделить все гамакообразные операторные подсхемы по исполнению.

Теорема 2. В операторной схеме $G(X,\Gamma,U)$ можно выделить все сцепленные и несцепленные операторные подсхемы с возвратом.

Теорема 3. В линейно перекрестной операторной схеме $G(X,\Gamma,U)$ можно выделить все линейные подсхемы.

Определение 19. Преобразование операторной схемы $G(X,\Gamma,U)$, при котором из подсхемы $G(X1,\Gamma,U)$ операторной схемы $G(X,\Gamma,U)$ получается полисемантический полиинформационный оператор с сохранением всех информационных связей и связей по передаче исполнения с операторной схемой $G(X,\Gamma,U)$, назовем синтезом.

Определение 20. Преобразование операторной подсхемы в несколько независимых подсхем с сохранением всех информационных связей и связей по передаче управления с операторной схемой назовем расщеплением

Определение 21. Подсхема операторной схемы, обрабатывающая группу входных данных независимо от дополнительной подсхемы, называется автономной по входным данным.

Теорема 1. В операторной схеме $G(X,\Gamma,U)$ можно выделить все гамакообразные операторные подсхемы по исполнению.

Теорема 2. В операторной схеме $G(X,\Gamma,U)$ можно выделить все сцепленные и несцепленные операторные подсхемы с возвратом.

Теорема 3. В линейно-перекрестной операторной схеме $G(X,\Gamma,U)$ можно выделить все линейные подсхемы.

2.3. Примеры программ с детерминированно-связанными модулями. Рассмотрим несколько программ с детерминированно-связанными модулями [3, 4].

Сортировка данных. Пусть имеются данные $a_1,\ldots,a_m$. Эти данные нужно пересортировать в неубывающей последовательности. Разместим последовательно исходные данные в k модулях. Программа сортировки перебором сравнений исходных данных следующая:

$i := 0$; $j := 0$; $t_1 := 1$; $t_2 := 1$;

цикл пока $t_1 = k$ цикл пока $j \neq c$ цикл пока $t_2 \neq k$

цикл пока $i \neq c$ установить $\leq (a_{t1,i}\ , a_{t2,i+1})$; $i := i+1$ конец

сохранить (t_2); $t_2 := t_2 + 1$; i := 0 конец $j := j + 1$; $t_2 := t_1$;

$i := j$ конец сохранить $(t_1\ )$; $t_1 := t_1 + 1$; $t_2 := t_1$; $i := 0$; $j := 0$;

$i := j$ конец сохранить (t_1); $t\ := t\ + 1$; $t\ ;= t\ $; $i := 0$; $j :+ 0$ конец

t_1, t_2 – счетчики модулей.

Оператор *установить* $\leq (a_{t1,i}, a_{t2,i+1})$ размещает значения указанной пары согласно отношению ≤. По адресу $a_{t1,i}$ размещается меньшее значение, а по адресу $a_{t2,i+1}$ – большее значение.

Оператор *сохранить* (t_1) отказывается от модуля с номером t_1 с сохранением его.

Структура программы с детерминированно-связанными модулями, определяющая использование операционного модуля и модулей данных, изображена на рис. 1.

Структура связи модуля PI_{k+1} с остальными модулями

Структура связи модуля PI_{k+1} с остальными модулями

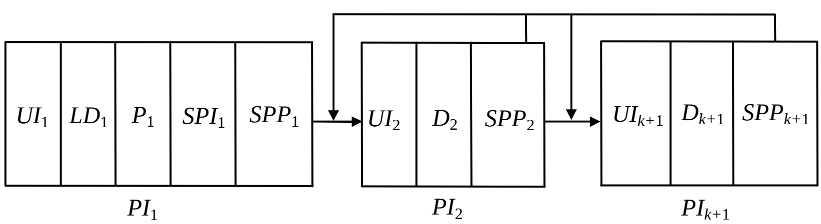

Рис. 1. Структура программы сортировки

Вывод модулей данных назначается в операционном модуле. Программа сортировки размещается в первом модуле, в остальных модулях размещаются данные. Модули модульной канонически связной программы связаны по номерам, кроме модуля с номером k + 1. Структура связи модуля k + 1 с остальными модулями с возвратами. Она определяется программой SPP_{k+1}: **если** $t_1 = 1$ **то** PI_2 **если** $t_1 = 2$ **то** PI_3 **… если** $t_1 = (k - 2)$ **то** PI_{k-1}.

Номера связи модуля и адрес программы связи модуля $\boldsymbol{k + 1}$ хранятся в управляющей информации.

На четыре часа обработки процессорами программы сортировки со случайными обращениями к модулям на современной ЭВМ замещение модулей на оперативной памяти занимает 20 ч. Результаты пользователь получает через 24 ч. При непрерывной обработке программ с детерминированно-связанными модулями на ЭВМ с упреждающим управлением памяти, пользователь получит результат через 16 ч.

Обработка банков данных. Пусть в банке данных имеются анкеты пяти миллиардов людей, в которых сообщены следующие данные: вес, рост, возраст, национальность, пол, социальное положение, специальность, должность, адрес и т.д. Нужно из пяти миллиардов людей выбрать всех, кто отвечает по своим анкетным данным эталону. Анкетные данные последовательно размещены в k модулях. В каждом модуле размещается q анкет. Программа выбора анкет по эталону следующая:

$i := 0$; ; $t_1 := POD\ j := 0$; $t_2 := 1$; $n := 0$; $t_3 := k + 1$;

цикл пока $t \neq k$ M**:** *цикл пока* $i \neq q$ *сравнить (эталон, анкета* (t_2, i)), *если да то записать* $(t_{1,j}, t_{2,i})$; $j := j + 1$

если $j > q$, *то сохранить* (t_1, t_3); $t_3 := t_3 + 1$; $n := n + 1$

если $n = 2$, то $t_1 := t_1 - 2$ *иначе,* $t_1 := t_1 + 1$; $j := 0$ на M

иначе $i := i + 1$ *конец освободить* (t_2); $t_2 := t_2 + 1$ *конец все*

t_1, t_2, t_3 – счетчики модулей.

Подпрограмма *сравнить (эталон, анкета* (t_2, i) сравнивает анкету i из модуля t_2 с эталоном.

Подпрограмма *записать* $(t_{1,j}, t)$ записывает анкету i из модуля t_1 в модуль t_2 по месту j.

Структура программы с детерминированно-связанными модулями, определяющая использование первого операционного модуля, k модулей анкет и с $k + 2$ по $k + p$ модулей эталонов, изображена на рис. 2.

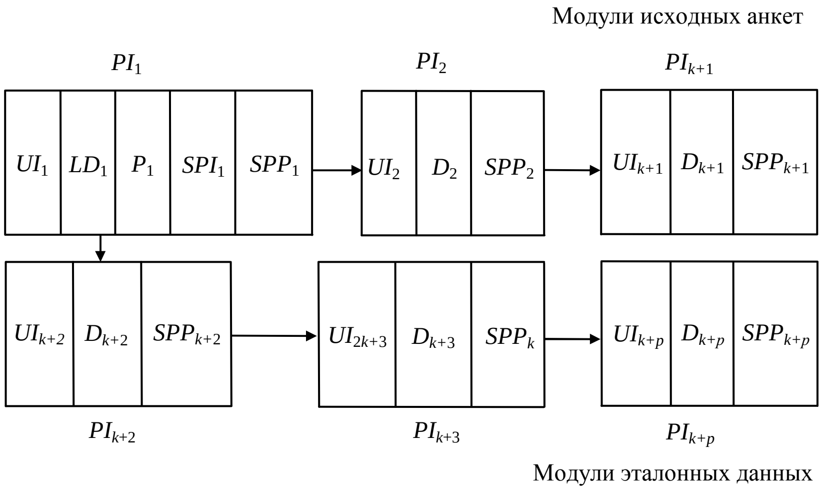

Рис. 2. Структура программы обработки банков данных

Программа обработки хранится в первом модуле. Далее идет последовательность модулей с анкетами. В модулях $PI_2,\ldots PI_{k+1}$ хранятся исходные анкеты. В модулях $PI_{k+2},\ldots PI_{k+p}$ хранятся эталонные анкеты. Модули связаны по номерам. Номера модулей хранятся в управляющей информации. В резидентном модуле общих данных накапливаются текущие исходные анкеты, совпадающие с эталоном.

Программа синтаксического разбора выражений. Нужно провести синтаксический контроль правильности записи выражения, находящегося в k модулях вводных данных, если понятие выражения определяется следующими синтаксическими правилами:

```
ВЫРАЖЕНИЕ ::= ТЕРМ I ТЕРМ + ВЫРАЖЕНИЕ
ТЕРМ ::= МНОЖИТЕЛЬ I МНОЖИТЕЛЬ * ТЕРМ
МНОЖИТЕЛЬ ::= ИМЯ I ЧИСЛО I В СКОБКАХ
ИМЯ  БУКВА I <ИМЯ><БУКВА> I <ИМЯ><ЦИФРА>
ЧИСЛО ::= ЦИФРА I <ЦИФРА>ЧИСЛО
В СКОБКАХ ::= (ВЫРАЖЕНИЕ)
```

Программа контроля синтаксической правильности выражения составляется из подпрограмм: `В СКОБКАХ, ЧИСЛО, ИМЯ, МНОЖИТЕЛЬ, ТЕРМ, ВЫРАЖЕНИЕ.`

Эти подпрограммы определяют правильность выражения соответствующей его части. Синтаксически правильные понятия выделяются из текста последовательно. Подпрограммы дают результат «да», если текст состоит из синтаксически правильных понятий. Анализ ведется до тех пор, когда счетчик модулей, будет удовлетворять условию $t = k + 1$.

При анализе используется модуль общих данных.

Синтаксическому анализу легко поддаются тексты за один проход. Структура программы с детерминированно-связанными модулями, определяющая использование первого операционного модуля и остальных текстовых модулей, изображена на рис. 3.

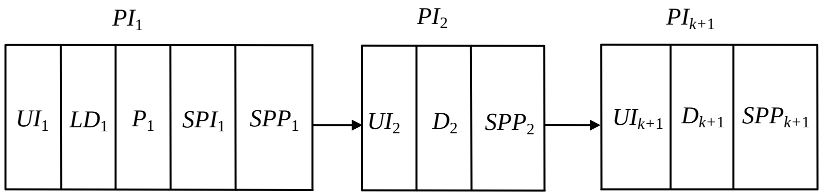

Рис. 3. Структура программы синтаксического разбора выражений.

Программа анализа располагается в первом модуле. Далее располагаются модули с входным текстом. Модули в последовательности связаны линейно по номерам. Номера модулей хранятся в управляющей информации.

3. Супер-ЭВМ с упреждающим управлением виртуальной памятью. Супер-ЭВМ с упреждающим управлением памятью реализуют параллельно-асинхронное взаимодействие актов использования ресурсов ЭВМ под управлением программ с детерминированно-связанными модулями. Супер-ЭВМ содержит новые устройства: процессор анализа связей между модулями программы, счетчики использования сегментов оперативной памяти модулями, процессор перемещения модулей по виртуальной памяти, процессор перемещения общих данных модулей (рис. 4). Процессор анализа проводит упреждающий анализ связей модулей программ с детерминированно-связанными модулями. Процессор анализа реализует процесс вычисления номера внешнего модуля перемещения на оперативную память по программе связи модуля, процесс корректировки значения счетчика использования сегментов оперативной памяти модулями программы, процесс запуска процессора перемещения модулей программы с внешней памяти на сегменты оперативной памяти. Процессор перемещения модулей реализует процессы перемещения модулей между устройствами внешней и оперативной памяти, обеспечивая наличие необходимых модулей на оперативной памяти по заявке процессора анализа связей. Процессор перемещения модулей реализует: процесс коммутации сегментов оперативной памяти с сегментами накопителя, процессы передачи и контроля информации модулей программы, процессы прерываний по передаче информации. Процессор перемещения общих данных реализует перемещение общих данных между модулями. Общие переменные имеют последовательности адресов перемещения их текущих значений. По последовательности адресов перемещения организуются потоки значений общих данных с доставкой их на место использования в модулях на оперативных сегментах. Обращение к модулям происходит по их номерам. Для модулей внешней памяти значения общих переменных переносятся в резидент общих данных. При замещении модуля, содержащего общие данные и их значения, переносятся из резидента. Активизация устройств ЭВМ осуществляется процессором управления по оттранслированной программе пользователя с детерминированно-связанными модулями.

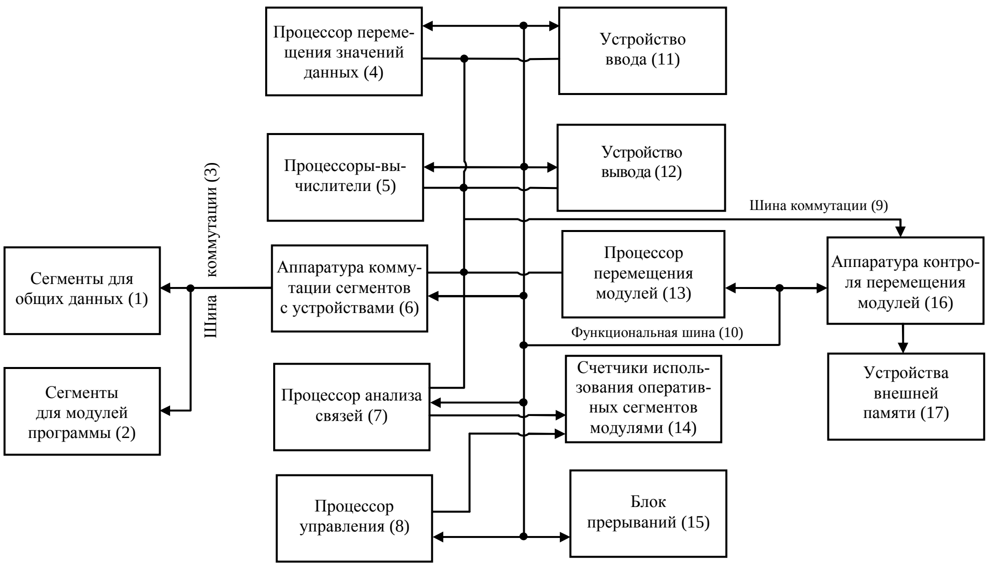

Рис. 4. Супер-ЭВМ с упреждающим управлением виртуальной памятью

Непрерывный вычислительный процесс начинается с загрузки начальных модулей программы с внешней памяти на сегменты оперативной памяти с перемещения значений общих данных и анализа связей между модулями. При загрузке модулей устанавливаются значения на счетчиках использования оперативных сегментов модулями. Для модулей данных, ввода, вывода значение счетчика равно единице. При загрузке модуля на сегмент оперативной памяти в управляющую информацию предшествующего по поиску модуля заносится номер оперативного сегмента. Затем процессор управления запускает процессоры обработки на обработку первого модуля. После выхода из каждого операционного модуля из значения соответствующего счетчика использования памяти вычитается единица и проверяется его значение на нуль. Для модулей данных значение счетчика обнуляется из операционного модуля по команде «обнулить счетчик».

Когда значение счетчика становится равным нулю и определен указатель на внешний модуль, тогда инициируется процессор перемещения модулей для замещения использованного модуля на оперативной памяти внешним модулем по указателю. После замещения указатель на внешний модуль обнуляется. Если в модуль на оперативной памяти не было записи, то он не переносится на внешнюю память. Процессор перемещения общих данных по адресным последовательностям переносит накопившиеся значения общих данных с резидента общих данных в заместивший модуль. Процессор анализа корректирует счетчик использования оперативной памяти заместившим модулем и определяет по программе связи заместившего модуля указатель на внешний модуль.

Процессор управления организует обработку, перемещение общих данных, анализ связей и замещение модулей. Он совмещает работу устройств над одним модулем за разные циклы обращений к оперативному сегменту.

Количество оперативных сегментов для непрерывной обработки программы с детерминированно-связанными модулями определяется в процессе ее трансляции или компиляции.

При технической реализации непрерывное присутствие текущих модулей в оперативной памяти достигается путем баланса скорости упреждающего обработку многоканального параллельного перемещения модулей с внешней памяти в оперативную со скоростью обработки информации из оперативной памяти.

ЭВМ с упреждающим управлением памятью обеспечивает непрерывный процесс обработки программы с детерминированно-связанными модулями упреждающим параллельным перемещением общих данных модулей и упреждающим параллельным замещением модулей на оперативной памяти через многоканальный обмен.

3.1. Поток значений общих данных. Готовые к последующей обработке значения общих данных перемещаются по модулям программы, находящимся в оперативной памяти. Для каждого значения общего данного *d* определяются последовательность использующих его модулей, места использования их в этих модулях и относительные моменты использования значений *d* в модулях. По множеству модулей использования *d* составляется дополнительное множество модулей, через которые перемещаются значения данного *d*.

Значения общих данных перемещаются по модулям, находящимся на сегментах оперативной памяти, динамически образуя поток данных.

Общие данные модулей, находящихся не на оперативной памяти, перемещаются в резидентные модули ROD. В резидентном модуле общих данных значения хранятся вместе с указателями перемещения. Значения, перемещаемые в один модуль, располагаются подряд. В начале последовательности указывается их количество. После записи новых значений в модуль общих данных перемещается его указатель свободного места (записи), если счетчик модуля общих данных не превышает допустимое число значений.

Значения снабжаются признаками перевычисления. Если признак принимает состояние неизменяемости, то значение перемещается во все используемые модули.

Значения помещаются в модуль общих данных в порядке их перемещения в модули, поступающие с внешней памяти на оперативную память. В модуле общих данных значения могут снабжаться несколькими указателями. После перемещения всех значений в модуль программы в нем устанавливается признак «перемещено», который указывает, что модуль готов к обработке.

Пусть имеется *k* модулей последовательности исполнения и *n* сегментов оперативной памяти. Пусть первый модуль имеет переменные. Для каждой переменной выпишем номера последующих модулей, в которых она используется. Для второго модуля выпишем все переменные, которых нет в первом модуле. Для каждой переменной выпишем номера последующих модулей, в которых она используется. Для последующих модулей аналогично выпишем последовательности использования переменных, которые не указаны в предыдущих модулях.

Для каждой переменной определим внешние модули. Выпишем последовательности номеров внешних использующих модулей. Переменные будут храниться в резидентном модуле общих данных согласно последовательной нумерации внешних модулей, использующих переменные.

3.2. Замещение модулей программы на оперативной памяти. Для хранения модулей программ используется внешняя память. Пусть каждый модуль размещается на отдельном сегменте. И пусть размеры сегмента и модуля определяются максимальной порцией обмена информацией между устройствами внешней памяти и оперативной памяти.

Сегмент оперативной памяти является локальным адресным пространством для модуля. Областью действия команд модуля, кроме команд процессора перемещения данных, является адресное пространство оперативного сегмента, на котором находится соответствующий модуль. Соответствие между сегментами оперативной памяти и модулями, находящимися на оперативных сегментах, устанавливается через ассоциативные регистры аппаратуры коммутации.

Если используются два сегмента оперативной памяти, то на одном сегменте хранится обрабатываемый модуль, на другом модуль готовится к обработке. Если используются три оперативных сегмента, то на одном сегменте происходит замещение обработанного модуля на модуль по внешней ссылке, на другом хранится обрабатываемый модуль, на третьем модуль готовится к обработке.

Замещение модулей на сегментах оперативной памяти определяется состоянием счетчиков использования сегментов модулями программы P и указателем внешнего модуля аппаратуры контроля перемещения модулей.

Разделение оперативной памяти на независимые сегменты делается для упреждающей подкачки модулей с внешней памяти на оперативную. Перемещаться между устройствами внешней и оперативной памяти одновременно могут несколько модулей.

Аппаратная реализация замещения модулей на оперативной памяти сводит практически время управления этим процессом к нулю в сравнении с временем их замещения и обработки.

Реализация полисемантических полиинформационных операторов (ППО) программы с детерминированно-связанными модулями осуществляется либо на ПЛИС-структурах, либо на сетевых ГРИД-системах с перестраиваемыми схемами. Для исполнения нескольких подряд ППО требуются две ПЛИС-структуры либо сетевые ГРИД-системы с перестраиваемыми схемами для непрерывной обработки программ.

Последовательность исполнения ППО на сетевых изменяемых структурах определяется после трансляции по последовательности исполнения детерминировано-связанных модулей программы.

Тело ППО хранится на внешней памяти отдельно от программы. Процедура загрузки тела ППО на сетевые структуры запускается механизмом упреждающей подкачки.

4. Конкурентные преимущества. Непрерывную обработку больших программ со случайными обращениями к модулям на существующих суперкомпьютерах реализуют путем увеличения оперативной памяти, построения ГРИД-систем, а также создания супер-ЭВМ с упреждающим управлением памятью.

1. Использование супер-ЭВМ с большой оперативной памятью для непрерывной обработки программ со случайным обращением к модулям является неэффективным из-за снижения производительности усложнением коммутации «процессоры–память». Такой подход является экономически невыгодным из-за существенного увеличения стоимости оперативной памяти.

2. Распределённые ГРИД-системы, состоящие из разновидности параллельных компьютеров со стандартными процессорами, устройствами хранения данных, блоками питания и т.д., подключенными к сети (локальной или глобальной) при помощи обычных протоколов, также не могут обеспечить непрерывную обработку больших программ со случайными обращениями к модулям. Таким образом, получаем практически те же вычислительные мощности для обработки больших программ, что и на современных суперкомпьютерах, работающих под управлением программ со случайными обращениями к их модулям. Как и в первом случае, обработка больших программ неэффективна и затратна.

3. Непрерывная обработка программ с детерминированно-связанными модулями на виртуальной памяти универсальной супер-ЭВМ с упреждающим управлением памятью сокращает время ожидания результата экспоненциально при обработке на виртуальной памяти целевой ЭВМ в сравнении с существующими супер-ЭВМ со случайным управлением памятью. Для исполняющихся за несколько суток больших программ, а также для обработки большого количества данных на виртуальной памяти в режиме реального времени это самый эффективный и экономически оптимальный метод. Время ожидания

результата сокращается за счет упреждающего перемещения модулей программы по виртуальной памяти, упреждающего перемещения данных по модулям программы, многопроцессорной параллельной обработки данных, готовых для вычислений.

5. Модельный вариант. Работоспособность целевой супер-ЭВМ проверена в режиме моделирования на интерпретаторе под модернизированную систему команд ЭВМ «Эльбрус». Интерпретатор для окончательной версии системы команд супер-ЭВМ «Эльбрус» был модернизирован в интерпретатор супер-ЭВМ с упреждающим замещением и непрерывной обработкой программ с детерминированно-связанными модулями на виртуальной памяти. Процедурный механизм системы команд супер-ЭВМ «Эльбрус» был модернизирован в механизм использования страниц оперативной памяти модулями программы. Упреждающая замена отработанных текущих модулей на страницах оперативной памяти на требуемые осуществлялась экстракодами обмена инструментальной ЭВМ БЭСМ-6.

Время получения результата на интерпретаторе целевой ЭВМ с упреждающим замещением сократилось на $T = t \times K$, где t – время одного обмена, K – количество замещений модулей на оперативной памяти. На 4 ч обработки программы сортировки со случайными обращениями к модулям на ЭВМ БЭСМ-6 замещение модулей на оперативной памяти заняло 20 ч. Результаты пользователь получал через 24 ч. При непрерывной обработке программ с детерминированно-связанными модулями пользователь получал результат через 4 ч.

6. Стратегия реализации супер-ЭВМ с упреждающим управлением памятью. Супер-ЭВМ с упреждающим управлением памятью можно реализовать модификацией отечественного суперкомпьютера «Ломоносов». Ее целесообразно провести с разработчиками компании Т-платформы. Суперкомпьютер «Ломоносов» производительностью 1 петафлопс, имеющий порядка 280 терабайт оперативной памяти и порядка 20 петабайт дискового пространства, нужно предварительно исследовать на эффективную модификацию в целевую супер-ЭВМ с упреждающим управлением памятью и многоканальным обменом виртуальной памяти.

Проект можно реализовать модификацией суперкомпьютера «Тяньхэ2». Ее целесообразно провести с китайскими разработчиками через государственный холдинг «Росэлектроника», в котором есть научно-исследовательское, опытно-конструкторское, производственное и маркетинговое подразделения. После превращения целевого суперкомпьютера в рыночный холдинг может создать международное акционерное общество с сохранением инфраструктуры для развития полученного инновационного рыночного продукта и реализации спроса на него.

Проект можно реализовать созданием консорциума в рамках национальной суперкомпьютерной технологической платформы, объединяющего фундаментальную исследовательскую деятельность, образовательную деятельность для подготовки кадров, прикладные научно-исследовательские и опытно-конструкторские работы и корпорации серийного производства целевой супер-ЭВМ с упреждающим управлением памятью и многоканальным обменом виртуальной памяти.

Библиографический список

1. Брындин Е.Г. Теоретические аспекты непрерывной обработки на виртуальной памяти // Информационные технологии. – М., 2009. – № 9. – С. 33–39.

2. Брындин Е.Г. Теоретические основы имитации мышления и непрерывной обработки на виртуальной памяти. – Новосибирск: Изд-во ИЦЕ; Томск: Изд-во ТПУ, 2011. – 235 с.

3. Брындин Е. Основы имитации мышления и непрерывной обработки программ. – Germany: LAP LAMBERT Academic Publishing, 2012. – 197 с.

4. Брындин Е.Г. Управление непрерывной обработкой программ на виртуальной памяти // Суперкомпьютеры. – 2014. – № 3. – С. 50–52.

References

1. Bryndin E.G. Teoreticheskie aspekty nepreryvnoi obrabotki na virtual'noi pamiati [Theoretical aspects of the continuous processing on the virtual storage]. Informatsionnye tekhnologii. Moscow, 2009, no. 9, pp. 33-39.

2. Bryndin E.G. Teoreticheskie osnovy imitatsii myshleniia i nepreryvnoi obrabotki na virtual'noi pamiati [Theoretical bases of simulation of thinking and the continuous processing on the virtual storage]. Novosibirsk: Izdatel'stvo Tomskogo politekhnicheskogo universiteta, 2011. 235 p.

3. Bryndin E. Osnovy imitatsii myshleniia i nepreryvnoi obrabotki program [Bases of simulation of thinking and the continuous processing of programs]. Germany: LAP LAMBERT Academic Publishing, 201. 197 p.

4. Bryndin E.G. Upravlenie nepreryvnoi obrabotkoi programm na virtual'noi pamiati [Control of the continuous processing of programs on the virtual storage]. Superkomp'iutery, 2014, no. 3, pp. 50-52.

Сведения об авторе

Брындин Евгений Григорьевич (Новосибирск, Россия) – директор НКО Исследовательского центра «Естествоинформатика» (630090, Новосибирск, ул. Терешковой, 10, офис. 15, e-mail: bryndin@ngs.ru; bryndin15@yandex.ru).

About the author

Bryndin Yevgeny Grigoryevich (Novosibirsk, Russian Federation) – the Director of the NKO Research Center "Estestvoinformatika" (630090, Russia, Novosibirsk, Tereshkova str., 10, Office 15, e-mail: bryndin@ngs.ru; bryndin15@yandex.ru).

Получено 20.02.2015