Math-Net.Ru

Общероссийский математический портал

Е. А. Аксенова, А. В. Соколов, Оптимальное управление двумя параллельными стеками в двухуровневой памяти, *Дискрет. матем.*, 2007, том 19, выпуск 1, 67–75

DOI: <https://doi.org/10.4213/dm9>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.89.185.249

26 апреля 2023 г., 12:10:44

Оптимальное управление двумя параллельными стеками в двухуровневой памяти

© 2007 г. Е. А. Аксенова, А. В. Соколов

Рассматривается задача управления двумя стеками в двухуровневой памяти. Предполагается, что вершины двух стеков растут навстречу друг другу в быстрой памяти, к которой разрешен доступ нескольких параллельных процессоров, и объем стеков превосходит объем быстрой памяти. В быстрой памяти хранятся только вершины стеков, а остальные части хранятся в памяти второго уровня. Если вершина одного из стеков стала пустой или стеки заполнили всю быструю память, то есть произошло переполнение стеков, то происходит обмен с памятью второго уровня так, что каждый раз устанавливается некоторое определенное состояние памяти, после чего начинается следующий этап работы. Рассматривается задача выбора этого состояния памяти в зависимости от вероятностей включения и исключения элементов стеков с тем, чтобы среднее время работы до следующего перераспределения памяти было максимальным.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проекты 01-01-00113 и 03-01-06415.

1. Введение

Понятие стека [1] используется при разработке программного и аппаратного обеспечения широкого класса задач (в данной работе рассматриваются только чистые стеки, в которых доступ к элементам, находящимся внутри стека запрещен). На практике (в стековых компьютерах и RISC-процессорах [2, 3]), при переполнении быстрой регистровой вершины стека, используется несколько методов.

Large Stack. Можно сделать стек большим и предполагать, что переполнений не будет, а если оно все же произошло, то совершать аварийное завершение работы.

Demand fed single element stack manager. В этой случае вершина стека реализована аппаратно, как циклический буфер, а продолжение стека находится в памяти. При переполнении или опустошении аппаратного стека перемещается один элемент (или несколько элементов) в память из буфера или наоборот.

Paging stack manager. В этом методе вершина реализована программно как часть специальной памяти. При опустошении вершины стека из памяти копируется половина буфера, а при переполнении нижняя половина буфера перемещается в память, а верхняя половина перемещается в начало буфера.

Barometer-Pointer Algorithm (см. [4]). Предлагались реализации стека, когда перемещения элементов между уровнями памяти происходят не при переполнениях или опустошениях, а при отклонении от оптимального состояния вершины (обычно считается, что это половина быстрого буфера) в фоновом режиме, то есть параллельно с обычным выполнением команд (аналог опережающей подкачки и откачки в страничной виртуальной памяти).

Associative cache. Кэш-память для стековых машин не дает преимуществ по сравнению с рассмотренными методами, так как ее использование сложнее для аппаратной реализации и в тоже время не учитывает специфики стека как структуры LIFO. Хотя, если в стек нужно помещать структуры данных переменного размера такие, как строки и структуры, использование кэш-памяти, как утверждается в [2], может быть целесообразно.

Если мы работаем с несколькими стеками (процессами) и общим ресурсом быстрой памяти, то можно показать, что оптимальным способом размещения стеков является такой, когда стеки разбиты на пары и растут навстречу друг другу (см. [5]). Задача построения математической модели такого способа распределения памяти была поставлена Д. Кнудом.

Пусть в памяти объема m расположены два стека, растущие навстречу друг другу. В этом случае место их встречи можно рассматривать как случайную величину. Пусть известно, что на каждом шаге с вероятностью $1 - p$ может произойти исключение элемента из одного из стеков, и с вероятностью p может произойти включение информации в один из стеков. Обозначим через $M(m, p)$ математическое ожидание случайной величины $\max(k_1, k_2)$, где k_1 и k_2 — длины стеков при встрече. Д. Кнут [1] поставил задачу, разработать математическую модель этого процесса и установить вид функции $M(m, p)$.

В [6–11] была построена математическая модель процесса в виде двумерного случайного блуждания в треугольной области с двумя отражающими экранами и одним поглощающим экраном.

В [6] был предложен алгоритм вычисления $M(m, p)$ для конечных значений m . Для решения задачи использовалась теория цепей Маркова с поглощающими экранами. Там же впервые была рассмотрена задача оптимального управления двумя стеками в двухуровневой памяти, которая рассматривалась также в [12–14].

В [7] решалась задача исследования $M(m, p)$ при $m \rightarrow \infty$ для $1/4 < p < 1/2$.

В [8–11] исследовалось асимптотическое поведение размеров стеков при переполнении и времени работы до переполнения памяти для $0 \leq p \leq 1/4$.

В данной работе рассмотрена задача оптимального управления двумя стеками в двухуровневой памяти. Предполагается, что вершины двух стеков растут навстречу друг другу в быстрой памяти, к которой разрешен доступ нескольких параллельных процессоров.

2. Постановка задачи

Пусть мы работаем с двумя растущими навстречу друг другу стеками, объем которых превосходит объем m быстрой памяти. В этом случае в быстрой памяти хранятся только вершины стеков, а остальные части хранятся в памяти второго уровня.

Если же вершина одного из стеков стала пустой или стеки заполнили всю быструю память, то есть произошло переполнение стеков, то происходит обмен с памятью второго

уровня так, что всегда приходим к некоторому заданному состоянию памяти, после чего начинается следующий этап работы.

Ставится задача, в какое состояние, в зависимости от заданных вероятностей включения и исключения элементов в стеки, следует переходить после обращения к памяти второго уровня, чтобы среднее время работы до следующего перераспределения памяти было максимально.

Обозначим x_1 текущую длину первого стека, а через x_2 текущую длину второго стека. Тогда в качестве математической модели процесса получаем случайное блуждание по целочисленной решетке в области $x_1 \geq 0, x_2 \geq 0$, где $x_1 + x_2 \leq m$ (не превосходит объем выделенной памяти). Параллельное выполнение операций подразумевает возможность одновременной работы с двумя стеками. В каждый момент времени могут произойти следующие операции над стеками:

- включение элемента в первый стек с вероятностью p_1 ,
- исключение элемента из первого стека с вероятностью q_1 ,
- включение элемента во второй стек с вероятностью p_2 ,
- исключение элемента из второго стека с вероятностью q_2 ,
- включение элементов в два стека одновременно с вероятностью p_{12} ,
- исключение элементов из двух стеков одновременно с вероятностью q_{12} ,
- включение в первый и одновременно исключение из второго с вероятностью pq_{12} ,
- исключение из первого и одновременно включение во второй с вероятностью qp_{12} ,
- с вероятностью r стеки не изменяют своей длины (только чтение или отсутствие операций), здесь $p_1 + q_1 + p_2 + q_2 + p_{12} + q_{12} + pq_{12} + qp_{12} + r = 1$.

3. Построение алгоритма решения

Рассмотрим примеры случайного блуждания по целочисленной решетке $x_1 \geq 0, x_2 \geq 0$, $x_1 + x_2 \leq m$ для $m = 1$ и $m = 2$:

Задача состоит в том, чтобы определить такое состояние $s_0 = (x_1^0, x_2^0)$, при выходе из которого время блуждания до поглощения на прямых $x_1 = -1, x_2 = -1$ и ломаной, которая задается уравнениями $x_1 + x_2 = m + 1$ и $x_1 + x_2 = m + 2$, было максимально.

Если пронумеровать состояния так, как показано на рис. 1, то есть сначала поглощающие из точки $(m, 0)$ (против часовой стрелки), а затем невозвратные (снизу вверх, справа налево), то получим конечную однородную поглощающую цепь Маркова [15].

Геометрически блуждание по решетке можно представить так: включение элемента в первый стек — сдвиг вправо по оси x_1 ; исключение элемента из первого стека — сдвиг влево по оси x_1 ; включение элемента во второй стек — сдвиг вверх по оси x_2 ; исключение элемента из второго стека — сдвиг вниз по оси x_2 ; включение элементов в два стека одновременно — сдвиг вправо вверх по диагонали; исключение элементов из двух стеков одновременно — сдвиг влево вниз по диагонали; включение в первый, исключение из второго одновременно — сдвиг вправо вниз по диагонали; исключение из первого, включение во второй одновременно — сдвиг влево вверх по диагонали.

Рис. 1.

Число невозвратных состояний вычисляется с помощью формулы суммы n членов арифметической прогрессии и равно $(m+1)(m+2)/2$. Общее число членов такой прогрессии равно $n = m+1$, первый член $a_1 = 1$, n -й член $a_n = m+1$. Число поглощающих состояний в такой цепи равно $4(m+2)$.

Теперь рассмотрим матрицу Q переходных вероятностей из невозвратных состояний в невозвратные. Если размер памяти равен m , то размерность матрицы Q равна $(m+1)(m+2)/2 \times (m+1)(m+2)/2$. Докажем теорему о структуре матрицы Q .

Теорема 1. Матрица Q при заданной нумерации и при данном размере памяти m имеет вид

$$Q = \begin{pmatrix} A_{m+1} & C_m & E_{m-1} & & & & 0 \\ B_{m+1} & A_m & C_{m-1} & E_{m-2} & & & \\ D_{m+1} & B_m & A_{m-1} & C_{m-2} & E_{m-3} & & \\ & D_m & B_{m-1} & A_{m-2} & C_{m-3} & E_{m-4} & \\ & & \dots & \dots & \dots & \dots & \\ & & & & & A_2 & C_1 \\ 0 & & & & & B_2 & A_1 \end{pmatrix},$$

где подматрица A_k размера $k \times k$ имеет вид

$$A_k = \begin{pmatrix} r & qp_{12} & & & \\ pq_{12} & r & qp_{12} & & \\ & pq_{12} & r & qp_{12} & \\ & & \ddots & \ddots & \ddots \\ & & & qp_{12} & r \end{pmatrix},$$

подматрица B_k размера $(k-1) \times k$ имеет вид

$$B_k = \begin{pmatrix} p_1 & p_2 & & & \\ & p_1 & p_2 & & \\ & & p_1 & p_2 & \\ & & & \ddots & \ddots \\ & & & & p_1 & p_2 \end{pmatrix},$$

подматрица C_k размера $k \times (k-1)$ имеет вид

$$C_k = \begin{pmatrix} q_1 & & & & \\ q_2 & q_1 & & & \\ & q_2 & q_1 & & \\ & & q_2 & \ddots & \\ & & & \ddots & q_1 \\ & & & & q_2 \end{pmatrix},$$

подматрица D_k размера $(k-2) \times k$ имеет вид

$$D_k = \begin{pmatrix} 0 & p_{12} & & & \\ & & p_{12} & & \\ & & & p_{12} & \\ & & & & \ddots \\ & & & & & p_{12} & 0 \end{pmatrix},$$

и подматрица E_k размера $k \times (k-2)$ имеет вид

$$E_k = \begin{pmatrix} 0 & & & & \\ q_{12} & & & & \\ & q_{12} & & & \\ & & \ddots & & \\ & & & q_{12} & \\ & & & & 0 \end{pmatrix}.$$

Доказательство. Пусть размер памяти $m = 1$ (см. рис. 1).

Тогда размер матрицы Q равен 3×3 и матрица имеет вид

$$Q_1 = \left(\begin{array}{cc|c} r & qp_{12} & q_1 \\ pq_{12} & r & q_2 \\ \hline p_1 & p_2 & r \end{array} \right).$$

Пусть размер памяти $m = 2$ (см. рис. 1).

Тогда размер матрицы Q равен 6×6 и матрица имеет вид

$$Q_2 = \left(\begin{array}{ccc|cc|c} r & qp_{12} & 0 & q_1 & 0 & 0 \\ pq_{12} & r & qp_{12} & q_2 & q_1 & q_{12} \\ 0 & pq_{12} & r & 0 & q_2 & 0 \\ \hline p_1 & p_2 & 0 & r & qp_{12} & q_1 \\ 0 & p_1 & p_2 & pq_{12} & r & q_2 \\ \hline 0 & p_{12} & 0 & p_1 & p_2 & r \end{array} \right).$$

Предположим, что для $m = n - 1$ выполняется утверждение теоремы. Матрица Q имеет размер $n(n + 1)/2 \times n(n + 1)/2$ и блочную структуру

$$Q_{n-1} = \begin{pmatrix} A_n & C_{n-1} & E_{n-2} & & & & 0 \\ B_n & A_{n-1} & C_{n-2} & E_{n-3} & & & \\ D_n & B_{n-1} & A_{n-2} & C_{n-3} & E_{n-4} & & \\ & D_{n-1} & B_{n-2} & A_{n-3} & C_{n-4} & E_{n-5} & \\ & & \dots & \dots & \dots & \dots & \\ & & & & & A_2 & C_1 \\ 0 & & & & & B_2 & A_1 \end{pmatrix}.$$

Проверим, что при $m = n$ выполняется утверждение теоремы. Число состояний в матрице Q_{n-1} увеличится на $n + 1$. Тогда число состояний в матрице будет равно $n(n + 1)/2 + (n + 1) = (n + 1)(n + 2)/2$. В матрицу Q_{n-1} при заданной нумерации добавятся подматрицы A_{n+1} размера $(n + 1) \times (n + 1)$, B_{n+1} размера $n \times (n + 1)$, C_n размера $(n + 1) \times n$, E_{n-1} размера $(n + 1) \times (n - 1)$ и D_{n+1} размера $(n - 1) \times (n + 1)$. Тем самым мы получаем матрицу Q_n размера $(n + 1)(n + 2)/2 \times (n + 1)(n + 2)/2$

$$Q_n = \left(\begin{array}{c|ccc} A_{n+1} & C_n & E_{n-1} & \\ \hline B_{n+1} & A_n & C_{n-1} & E_{n-2} & & & 0 \\ D_{n+1} & B_n & A_{n-1} & C_{n-2} & E_{n-3} & & \\ & D_n & B_{n-1} & A_{n-2} & C_{n-3} & E_{n-4} & \\ & & \dots & \dots & \dots & \dots & \\ & & & & & A_2 & C_1 \\ & & & & & B_2 & A_1 \\ \hline & 0 & & & & & \end{array} \right).$$

Теорема доказана.

Можно заметить, что подматрицы матрицы Q характеризуют определенные операции над стеками. Подматрица A_k характеризует операции включения в один из стеков и исключение из другого одновременно, а также возможность остаться на месте, подматрица B_k — операции включения в один из стеков, подматрица C_k — операции исключения из какого-то стека, подматрица D_k — операции включения в оба стека одновременно, подматрица E_k — операции исключения из двух стеков одновременно.

Найдем фундаментальную матрицу

$$N = (I - Q)^{-1}$$

(см. [15]), где I — единичная матрица. Элемент N_{ij} равен среднему числу единиц времени, которое процесс находился в состоянии j , если блуждание началось из состояния i . Значит, чтобы найти математическое ожидание $E(t)$ числа шагов до перераспределения, нужно найти суммы элементов строк фундаментальной матрицы, добавляя к каждой единицу, так как для того, чтобы произошло перераспределение, нужен еще один шаг для переполнения или опустошения. Тогда для того, чтобы найти оптимальное состояние $s_0 = (x_1^0, x_2^0)$, необходимо из полученных сумм элементов матрицы Q выбрать максимальную. Таким образом получаем максимальное среднее время блуждания до перераспределения. Номер состояния, соответствующий этому максимальному времени определяется однозначно с помощью матрицы Q , а каждому номеру состояния соответствует определенное состояние s_0 памяти (см. рис. 1).

Таблица 1.

r	p_1	q_1	p_2	q_2	x_1^0	x_2^0	T
0	0,25	0,25	0,25	0,25	3	3	20,896
0	0,1	0,1	0,4	0,4	2	4	22,871
0	0,4	0,4	0,1	0,1	4	2	22,871
0	0,2	0,2	0,3	0,3	3	3	20,972
0	0,3	0,3	0,2	0,2	3	3	20,972
0	0,1	0,3	0,2	0,4	4	4	15,489152
0	0,2	0,4	0,3	0,1	5	1	17,857359
0,2	0,1	0,1	0,3	0,3	2	4	27,345371
0,2	0,3	0,1	0,3	0,1	1	1	18,843216
0,4	0,1	0,1	0,2	0,2	3	3	34,51078
0,4	0,1	0,2	0,1	0,2	4	4	26,998907
0,1	0,225	0,225	0,225	0,225	3	3	23,107
0,2	0,2	0,2	0,2	0,2	3	3	25,87
0,3	0,175	0,175	0,175	0,175	3	3	29,423
0,4	0,15	0,15	0,15	0,15	3	3	34,16
0,5	0,125	0,125	0,125	0,125	3	3	40,792

4. Численные результаты

Для решения задачи были разработаны алгоритм и программа, определяющие оптимальное состояние $s_0 = (x_1^0, x_2^0)$. Некоторые результаты вычислений представлены в таблицах 1, 2, 3, 4 для $m = 10$. В таблицах столбцы с обозначением вероятностей соответствуют заданным вероятностям блуждания, в столбцах x_1^0 и x_2^0 находится оптимальное число единиц памяти, которое должно быть выделено первому и второму стеку, в столбце T находится среднее время блуждания (среднее время работы со стеками между двумя последовательными перераспределениями) при заданных вероятностях блуждания и данным оптимальном состоянии вершин стеков.

В табл. 1 представлены результаты для случая, когда

$$p_{12} = qp_{12} = pq_{12} = q_{12} = 0,$$

то есть когда не допускается параллельное выполнение операций.

В табл. 2 представлены результаты, когда

$$p_1 = q_1 = p_2 = q_2 = 0,$$

то есть когда происходит только параллельное выполнение операций. Здесь представлены такие значения параметров, которые имеют прозрачную интуитивную интерпретацию. Например, если вероятность включения в оба стека одновременно $p_{12} = 1$, а все остальные равны нулю, то ясно, что оптимальное состояние $s_0 = (0, 0)$.

В табл. 3 представлены результаты, когда изменяется только r , а остальные вероятности равны между собой.

В табл. 4 представлены результаты для случая, когда $r = 0$, а остальные вероятности изменяются.

Полученные данные позволяют сделать вывод, что в симметричных случаях $x_1^{(0)} \approx x_2^{(0)} \approx m/3$.

Анализ в целом всей модели сложный, так как трудно отслеживать поведение оптимальной точки при разных значениях вероятностей.

Таблица 2.

r	p_{12}	qp_{12}	pq_{12}	q_{12}	x_1^0	x_2^0	T
0	1	0	0	0	0	0	7.00
0	0	1	0	0	10	0	12.00
0	0	0	1	0	0	10	12.00
0	0	0	0	1	5	5	7.00
0	0	0,5	0,5	0	5	5	37.00
0	0,5	0	0	0,5	3	3	3.00
0	0,1	0,3	0,2	0,4	5	3	10,144488
0	0,2	0,4	0,3	0,1	4	2	13.68349
0,2	0,1	0,1	0,3	0,3	3	5	12.561406
0,2	0,3	0,1	0,3	0,1	1	3	12.828635
0,4	0,1	0,1	0,2	0,2	3	5	18.376192
0,4	0,1	0,2	0,1	0,2	5	3	18.37619
0,1	0,225	0,225	0,225	0,225	3	3	13.846
0,2	0,2	0,2	0,2	0,2	3	3	15.451
0,3	0,175	0,175	0,175	0,175	3	3	17.516
0,4	0,15	0,15	0,15	0,15	3	3	20,268
0,5	0,125	0,125	0,125	0,125	3	3	24.121

Таблица 3.

r	$p_1 = q_1 = p_2 = q_2 = p_{12} = qp_{12} = pq_{12} = q_{12}$	x_1^0	x_2^0	T
0	0,125	3	3	14.83006
0,1	0,1125	3	3	16.366
0,2	0,1	3	3	18.287567
0,3	0,0875	3	3	20,757233
0,4	0,075	3	3	24.65.00
0,5	0,0625	3	3	28.660124
0,9	0,0125	3	3	139.00

5. Заключение

На практике вероятностные характеристики стеков зависят от многих факторов. В большинстве приложений в начале работы более вероятно включение в стек, в середине работы вероятности включения и исключения равны, а в конце наиболее вероятно исключение из стека, так как начинается и заканчивается работа обычно пустым стеком. Для реализации оптимальных алгоритмов управления стеками необходимо провести исследование поведения стека для различных входных данных.

Таблица 4.

r	p_1	q_1	p_2	q_2	p_{12}	qp_{12}	pq_{12}	q_{12}	x_1^0	x_2^0	T
0	0,1	0,2	0,1	0,2	0,1	0,1	0,1	0,1	4	3	14.467404
0	0,2	0,1	0,2	0,1	0,2	0,1	0,1	0	1	1	12.029736
0	0,2	0,1	0,2	0,1	0	0,1	0,1	0,2	3	4	15.160632
0	0,1	0,1	0,1	0,1	0,1	0,2	0,2	0,1	3	3	15.129286

Список литературы

1. Кнут Д., *Искусство программирования, т. 1: Основные алгоритмы*. Вильямс, Москва, 2000.
2. Koopman P., *Stack computers*. Ellis Horwood, Chichester, 1989.
3. Катеванис М. Г. Х., Секуин К. Х., Паттерсон Д. А., Шербурн Р. У., *Эффективные архитектуры для СБИС-компьютеров. Электроника СБИС. Проектирование микроструктур*. Мир, Москва, 1989.
4. Stanley T., Wedig R., A performance analysis of automatically managed top of stack buffers. In: *Proc. 14th Annual International Symposium on Computer Architecture*. IEEE Computer Society Press, Pittsburgh, 1987, pp. 272–281.
5. Соколов А. В., *Математические модели и алгоритмы оптимального управления динамическими структурами данных*. Изд-во Петрозаводского госуниверситета, Петрозаводск, 2002.
6. Соколов А. В., *О распределении памяти для двух стеков. Автоматизация эксперимента и обработки данных*. КФ АН СССР, Петрозаводск, 1980.
7. Yao A. C., An analysis of a memory allocation scheme for implementing stacks. *SIAM J. Comput.* (1981) **10**, 398–403.
8. Flajolet P., The evolution of two stacks in bounded space and random walks in a triangle. *Lecture Notes Computer Sci.* (1986) **223**, 325–340.
9. Louchard G., Schott R., Probabilistic analysis of some distributed algorithms. *Lecture Notes Computer Sci.* (1990) **431**, 239–253.
10. Louchard G., Schott R., Tolly M., Zimmermann P., Random walks, heat equation and distributed algorithms. *J. Comput. Appl. Math.* (1994) **53**, 243–274.
11. Maier R. S., Colliding stacks: a large deviation analysis. *Random Structures and Algorithms* (1991) **2**, 379–421.
12. Sokolov A. V., Mathematical models and optimal algorithms of dynamic data structure control. *Lecture Notes Computer Sci.* (2001) **2138**, 416–419.
13. Аксенова Е. А., Волкова О. В., Лазутина А. А., Соколов А. В., Методы оптимального управления стеками в двухуровневой памяти. *Труды ИПМИ КарНЦ РАН* (2002) **3**, 127–152.
14. Sokolov A. V., Optimization strategies of stack control. In: *Recent Advances in Java Technology. Theory, Application, Implementation*. Computer Sci. Press, Trinity College, Dublin, pp. 193–203.
15. Кемени Дж., Снелл Дж., *Конечные цепи Маркова*. Наука, Москва, 1970.

Статья поступила 20.01.2004.