УДК 004.942
doi:10.15217/issn1684-8853.2015.5.65

ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ ДВУМЯ ПАРАЛЛЕЛЬНЫМИ FIFO-ОЧЕРЕДЯМИ НА БЕСКОНЕЧНОМ ВРЕМЕНИ

Е. А. Барковский[а], *аспирант*
А. В. Соколов[а, б], *доктор физ.-мат. наук, профессор*
[а]*Институт прикладных математических исследований Карельского научного центра Российской академии наук, Петрозаводск, РФ*
[б]*Петрозаводский государственный университет, Петрозаводск, РФ*

Введение: *FIFO-очередь является очень распространенной структурой данных: ее применяют во многих аппаратных и программных приложениях. При разработке различных сетевых устройств и встроенных операционных систем требуется работа с несколькими FIFO-очередями, расположенными в общем пространстве памяти. Также существуют архитектуры многоядерных процессоров, где каждому ядру выделено две FIFO-очереди. Целью исследования является построение и анализ математической модели процесса работы с двумя последовательными циклическими FIFO-очередями в общей памяти, когда на нечетном шаге происходят операции включения элементов в одну из очередей, а на четном шаге — исключения (возможно как последовательное, так и параллельное выполнение операций).* ***Результаты:*** *сформулирована задача оптимального разбиения общей памяти FIFO-очередей как задача целочисленного программирования, где функция критерия оптимальности задается алгоритмически. Построены математическая и имитационная модели этого процесса для двух очередей и проведены численные эксперименты, основывающиеся на теоретических данных. Математическая модель представлена в виде случайного блуждания по двухмерной целочисленной решетке, имеющей отражающие экраны, т. е. мы имеем дело с регулярной однородной марковской цепью. Критерием оптимальности является минимальная средняя доля потерянных при переполнении элементов очередей. Особенностью данного исследования является специфическое выполнение операций над очередями: включение и исключение элементов происходит в зависимости от шага (сделаны поправки для сохранения качеств однородности и регулярности цепи) и выполнение операции возможно параллельно.* ***Практическая значимость:*** *с помощью разработанной модели можно найти оптимальное разделение ограниченной общей памяти для повышения стабильности работы системы. Предложенные модели, алгоритмы и разработанный программный комплекс могут применяться при проектировании сетевых устройств, например маршрутизаторов, где потери пакетов являются допустимой, но нежелательной ситуацией. Разделяя общую память для очередей оптимально, мы теряем меньше пакетов, и, как следствие, данные доставляются быстрее.*

Ключевые слова *— структуры данных, FIFO-очереди, случайные блуждания, регулярные марковские цепи.*

Введение

Во многих приложениях, например при разработке различных сетевых устройств и встроенных операционных систем, требуется работа с несколькими FIFO-очередями, расположенными в общем пространстве памяти. Механизм страничной виртуальной памяти здесь не используется, и вся работа происходит в нескольких пулах оперативной памяти. Количество очередей в таких устройствах может достигать нескольких сотен и тысяч, а в будущем, по экспертным оценкам, может достигнуть нескольких миллионов. Для представления FIFO-очередей применяют различные программные или аппаратные решения [1–3].

Отметим также, что среди архитектур многоядерных процессоров есть и такие, где отсутствует кэш-память. Например, в архитектуре AsAP-II каждое ядро имеет два FIFO-буфера, а в архитектуре SEAforth — два стека для хранения данных и адресов возвратов [4]. В этих архитектурах очереди и стеки реализованы циклически и раздельно с возможностью потери элементов при переполнении. Мы же исследуем ситуации, когда для хранения нескольких структур данных используется общая память, что в ряде случаев позволяет снизить потери элементов при переполнении.

Д. Кнут поставил задачу [1] построения и анализа математической модели работы с двумя стеками, растущими навстречу друг другу. Для описания этого процесса были построены модели в виде случайного блуждания в треугольнике [5–10]. В работах [11–14] предлагались модели для последовательного, связанного и страничного способов представления нескольких FIFO-очередей в памяти одного уровня. В этих моделях предполагается, что на каждом шаге дискретного времени с заданными вероятностями происходят некоторые операции со структурами данных. Время выполнения операций — это не случайная величина, а константа, поэтому фиксированным является и шаг времени.

В работе [2] приведены результаты имитационных экспериментов и поставлена задача построить математическую модель процесса работы с несколькими FIFO-очередями в общей памяти, когда операции с очередями выполняются по несколько другому принципу. В данной схеме работы на нечетном шаге допускаются операции включения элементов в одну из n очередей с равными вероятностями, а на четном шаге — опера-

ции исключения элементов из очередей с равными вероятностями. Исключение из пустой очереди не приводит к завершению работы.

Также ставилась задача [2] определить вероятность (как функцию от n и j) того, что очередь, выбранная для операции на j-м шаге, будет пустой, и вычислить математическое ожидание количества элементов в очередях после j операций. В данной задаче не рассматривался конкретный способ представления очередей в памяти, т. е. предполагалось, что очереди могут быть неограниченной длины, что на практике невыполнимо. Эта задача была решена в работе [14].

В данной работе мы предлагаем математическую модель и решаем задачу оптимального разбиения общей памяти ограниченного объема для двух FIFO-очередей в случае последовательного циклического представления очередей. Операции с очередями выполняются по принципу, предложенному выше, и выполнение операций (с заданными вероятностями) может происходить как последовательно, так и параллельно.

В качестве критерия оптимальности рассматривается минимальная доля потерянных элементов при бесконечном времени работы очередей. Эту величину разумно минимизировать, когда переполнение очереди является не аварийной, а стандартной ситуацией (здесь мы подчеркиваем, что в некоторых приложениях при переполнении очереди работа программы заканчивается, и тогда в качестве критерия оптимальности надо рассматривать максимальное среднее время до переполнения памяти). Если очередь занимает всю предоставленную ей память, то все последующие элементы, поступающие в нее, отбрасываются до тех пор, пока не появится свободная память (т. е. пока не произойдет исключение элемента из очереди). Такая схема применяется, например, в работе сетевых маршрутизаторов [3] в том случае, когда по мере увеличения трафика очередь на исходящем интерфейсе маршрутизатора заполняется пакетами. Такое поведение маршрутизатора называется «сбросом хвоста». Потери пакетов приводят к нежелательному результату, поэтому число таких ситуаций необходимо свести к минимуму.

Математическая модель

Пусть в памяти размером в m единиц мы работаем с двумя последовательными циклическими FIFO-очередями с элементами фиксированного размера в одну условную единицу.

Для последовательного представления каждой очереди выделим некоторое количество единиц памяти из общего объема, равного m единиц. Пусть s — количество единиц памяти, выделенных первой очереди, тогда $(m - s)$ — количество единиц памяти, выделенных второй очереди.

Операции, производимые с очередями, выполняются по следующей схеме: на нечетном шаге происходит операция включения элемента в одну из очередей, на четном шаге — операция исключения элемента из какой-либо очереди, причем известны некоторые вероятностные характеристики операций, производимых с очередями. Пусть p_1 и p_2 — вероятности включения элемента в первую и вторую очереди, соответственно p_{12} — вероятность одновременного включения в обе очереди; q_1 и q_2 — вероятности исключения элемента из первой и второй очередей, соответственно q_{12} — вероятность одновременного исключения из обеих очередей.

Поскольку построенная на основе такой постановки задачи марковская цепь [15] не будет регулярной и однородной, два последовательных шага объединяем в один, а также вводим в рассмотрение вероятности выполнения операций, не изменяющих длины очередей. Например, чтение: r_1 — вероятность выполнения операции на нечетном шаге и r_2 — на четном, при этом $r_1 \neq 0$, $r_2 \neq 0$. Соответственно: $p_1 + p_2 + p_{12} + r_1 = 1$, $q_1 + q_2 + q_{12} + r_2 = 1$.

Тогда состояния на каждом шаге определяются наступлением одной из следующих комбинаций событий:

1) включение в первую, исключение из второй очереди с вероятностью $p_1 q_2$;

2) включение во вторую, исключение из первой очереди с вероятностью $p_2 q_1$;

3) включение в первую очередь с вероятностью $p_1 r_2 + p_{12} q_2$;

4) включение во вторую очередь с вероятностью $p_2 r_2 + p_{12} q_1$;

5) включение параллельно в обе очереди с вероятностью $p_{12} r_2$;

6) исключение из первой очереди с вероятностью $q_1 r_1 + p_2 q_{12}$;

7) исключение из второй очереди с вероятностью $q_2 r_1 + p_1 q_{12}$;

8) исключение параллельно из обеих очередей с вероятностью $q_{12} r_1$;

9) выполнение над очередями сохраняющих их состояние противоположных операций с вероятностью $r_1 r_2 + p_1 q_1 + p_2 q_2 + p_{12} q_{12}$, при этом сумма вероятностей всех событий равна 1.

Целью исследования является определение оптимального распределения памяти между очередями, когда в качестве критерия оптимальности рассматривается минимальная средняя доля потерянных при переполнении элементов очередей. По закону больших чисел для регулярных цепей Маркова [15] это эквивалентно нахождению решения, доставляющего минимум значению вероятности переполнения памяти на бесконечном промежутке времени.

Обозначим через x и y текущие длины первой и второй очередей соответственно. В качестве

математической модели рассматриваем случайное блуждание в двумерном пространстве по целочисленной решетке в области $0 \le x \le s + 2$, $0 \le y \le m - s + 2$ (рис. 1).

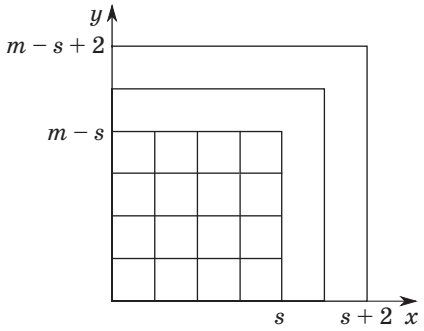
Прямые $x = s + 1$, $y = m - s + 1$ образуют первый отражающий экран; попадая на эти прямые, мы будем находиться на них до тех пор, пока не произойдет исключение элемента из очереди. Прямые $x = s + 2$, $y = m - s + 2$ образуют второй отражающий экран, который определен для случаев включения элемента в заполненную очередь и немедленного исключения элемента из этой же очереди. Введением данного экрана учитывается произошедшая потеря элемента, очередь формально переходит на экран, а фактически в область $0 \le x < s$, $0 \le y < m - s$, конкретно на прямые $x = s - 1$ или $y = m - s - 1$.

Определим схему переходов между состояниями. Пусть (x, y) — текущее состояние процесса, тогда блуждание в области $0 \le x \le s + 1$, $0 \le y \le m - s + 1$ (включает в себя и первый отражающий экран) можно описать следующим образом:

$$(x,y) \xrightarrow{r_1 r_2} (x',y') = \begin{cases} (x,y), & 0 \le x \le s, 0 \le y \le m-s \\ (x,y-1), & 0 \le x \le s, y = m-s+1 \\ (x-1,y), & x = s+1, 0 \le y \le m-s \\ (x-1,y-1), & x = s+1, y = m-s+1 \end{cases};$$

$$(x,y) \xrightarrow{p_1 r_2} (x',y') = \begin{cases} (x,y), & x = s+1, 0 \le y \le m-s \\ (x,y-1), & x = s+1, y = m-s+1 \\ (x+1,y), & 0 \le x \le s, 0 \le y \le m-s \\ (x+1,y-1), & 0 \le x \le s, y = m-s+1 \end{cases};$$

$$(x,y) \xrightarrow{p_1 q_1} (x',y') = \begin{cases} (x,y), & 0 \le x < s, 0 \le y \le m-s \\ (x,y-1), & 0 \le x < s, y = m-s+1 \\ (x+1,y), & x = s+1, 0 \le y \le m-s \\ (x+1,y-1), & x = s+1, y = m-s+1 \\ (x+2,y), & x = s, 0 \le y \le m-s \\ (x+2,y-1), & x = s, y = m-s+1 \end{cases};$$



■ ***Рис. 1.*** Область блуждания

$$(x,y) \xrightarrow{p_1 q_2} (x',y') = \begin{cases} (x,y), & x = s+1, y = 0 \\ (x,y-1), & x = s+1, 0 < y \le m-s \\ (x,y-2), & x = s+1, y = m-s+1 \\ (x+1,y), & 0 \le x \le s, y = 0 \\ (x+1,y-1), & 0 \le x \le s, 0 < y \le m-s \\ (x+1,y-2), & 0 \le x \le s, y = m-s+1 \end{cases};$$

$$(x,y) \xrightarrow{p_1 q_{12}} (x',y') = \begin{cases} (x,y), & 0 \le x < s, y = 0 \\ (x,y-1), & 0 \le x < s, 0 < y \le m-s \\ (x,y-2), & 0 \le x < s, y = m-s+1 \\ (x+1,y), & x = s+1, y = 0 \\ (x+1,y-1), & x = s+1, 0 < y \le m-s \\ (x+1,y-2), & x = s+1, y = m-s+1 \\ (x+2,y), & x = s, y = 0 \end{cases};$$

$$(x,y) \xrightarrow{p_2 r_2} (x',y') = \begin{cases} (x,y), & 0 \le x \le s, y = m-s+1 \\ (x,y+1), & 0 \le x \le s, 0 \le y \le m-s \\ (x-1,y), & x = s+1, y = m-s+1 \\ (x-1,y+1), & x = s+1, 0 \le y \le m-s \end{cases};$$

$$(x,y) \xrightarrow{p_2 q_1} (x',y') = \begin{cases} (x,y), & x = 0, y = m-s+1 \\ (x,y+1), & x = 0, 0 \le y \le m-s \\ (x-1,y), & 0 \le x \le s, y = m-s+1 \\ (x-1,y+1), & 0 \le x \le s, 0 \le y \le m-s \\ (x-2,y), & x = s+1, y = m-s+1 \\ (x-2,y+1), & x = s+1, 0 \le y \le m-s \end{cases};$$

$$(x,y) \xrightarrow{p_2 q_2} (x',y') = \begin{cases} (x,y), & 0 \le x \le s, 0 \le y < m-s \\ (x,y+1), & 0 \le x \le s, y = m-s+1 \\ (x-1,y), & x = s+1, 0 \le y < m-s \\ (x-1,y+1), & x = s+1, y = m-s+1 \\ (x,y+2), & 0 \le x \le s, y = m-s \\ (x-1,y+2), & x = s+1, y = m-s \end{cases};$$

$$(x,y) \xrightarrow{p_2 q_{12}} (x',y') = \begin{cases} (x,y), & x = 0, 0 \le y < m-s \\ (x,y+1), & x = 0, y = m-s+1 \\ (x,y+2), & x = 0, y = m-s \\ (x-1,y), & 0 < x \le s, 0 \le y < m-s \\ (x-1,y+1), & 0 < x \le s, y = m-s+1 \\ (x-1,y+2), & 0 < x \le s, y = m-s \\ (x-2,y), & x = s+1, 0 \le y < m-s \\ (x-2,y+1), & x = s+1, y = m-s+1 \\ (x-2,y+2), & x = s+1, y = m-s \end{cases};$$

$$(x,y) \xrightarrow{p_{12} r_2} (x',y') = \begin{cases} (x,y), & x = s+1, y = m-s+1 \\ (x,y+1), & x = s+1, 0 \le y \le m-s \\ (x+1,y), & 0 \le x \le s, y = m-s+1 \\ (x+1,y+1), & 0 \le x \le s, 0 \le y \le m-s \end{cases};$$

$$(x,y)\overset{p_{12}q_1}{\rightarrow}(x',y')=\begin{cases}(x,y), & 0\le x<s,\ y=m-s+1\\ (x,y+1), & 0\le x<s,\ 0\le y\le m-s\\ (x+1,y), & x=s+1,\ y=m-s+1\\ (x+1,y+1), & x=s+1,\ 0\le y\le m-s\\ (x+2,y), & x=s,\ y=m-s+1\\ (x+2,y+1), & x=s,\ 0\le y\le m-s\end{cases};$$

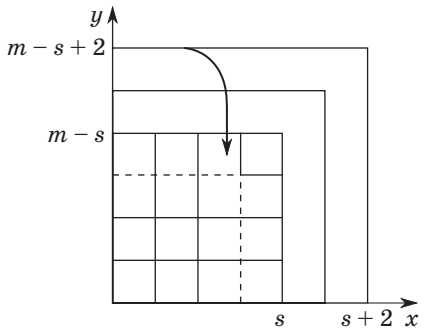
$$(x,y)\overset{p_{12}q_2}{\rightarrow}(x',y')=\begin{cases}(x,y), & x=s+1,\ 0\le y<m-s\\ (x,y+1), & x=s+1,\ y=m-s+1\\ (x,y+2), & x=s+1,\ y=m-s\\ (x+1,y), & 0\le x\le s,\ 0\le y<m-s\\ (x+1,y+1), & 0\le x\le s,\ y=m-s+1\\ (x+1,y+2), & 0\le x\le s,\ y=m-s\end{cases};$$

$$(x,y)\overset{p_{12}q_{12}}{\rightarrow}(x',y')=\begin{cases}(x,y), & 0\le x<s,\ 0\le y<m-s\\ (x,y+1), & 0\le x<s,\ y=m-s+1\\ (x,y+2), & 0\le x<s,\ y=m-s\\ (x+1,y), & x=s+1,\ 0\le y<m-s\\ (x+1,y+1), & x=s+1,\ y=m-s+1\\ (x+1,y+2), & x=s+1,\ y=m-s\\ (x+2,y), & x=s,\ 0\le y<m-s\\ (x+2,y+1), & x=s,\ y=m-s+1\\ (x+2,y+2), & x=s,\ y=m-s\end{cases};$$

$$(x,y)\overset{q_1r_1}{\rightarrow}(x',y')=\begin{cases}(x,y), & x=0,\ 0\le y\le m-s\\ (x,y-1), & x=0,\ y=m-s+1\\ (x-1,y), & 0<x\le s,\ 0\le y\le m-s\\ (x-1,y-1), & 0<x\le s,\ y=m-s+1\\ (x-2,y), & x=s+1,\ 0\le y\le m-s\\ (x-2,y-1), & x=s+1,\ y=m-s+1\end{cases};$$

$$(x,y)\overset{q_2r_1}{\rightarrow}(x',y')=\begin{cases}(x,y), & 0\le x\le s,\ y=0\\ (x,y-1), & 0\le x\le s,\ 0<y\le m-s\\ (x,y-2), & 0\le x\le s,\ y=m-s+1\\ (x-1,y), & x=s+1,\ y=0\\ (x-1,y-1), & x=s+1,\ 0<y\le m-s\\ (x-1,y-2), & x=s+1,\ y=m-s+1\end{cases};$$

$$(x,y)\overset{q_{12}r_1}{\rightarrow}(x',y')=\begin{cases}(x,y), & x=0,\ y=0\\ (x,y-1), & x=0,\ 0\le y\le m-s\\ (x,y-2), & x=0,\ y=m-s+1\\ (x-1,y), & 0\le x\le s,\ y=0\\ (x-1,y-1), & 0\le x\le s,\ 0\le y\le m-s.\\ (x-1,y-2), & 0\le x\le s,\ y=m-s+1\\ (x-2,y), & x=s+1,\ y=0\\ (x-2,y-1), & x=s+1,\ 0\le y\le m-s\\ (x-2,y-2), & x=s+1,\ y=m-s+1\end{cases}$$



■ ***Рис. 2.*** Состояния в области блуждания, соответствующие второму экрану

Переходы со второго отражающего экрана (рис. 2):

1) при $y = m–s + 2$ переходы из состояния (x, y) соответствуют переходам из состояния (x', y'), где $x' = x$, $y' = y–3$;

2) при $x = s + 2$ переходы из состояния (x, y) соответствуют переходам из состояния (x', y'), где $x' = x–3$, $y' = y$;

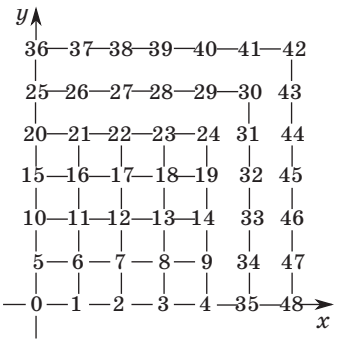
3) при $x = s + 2$ и $y = m – s + 2$ переходы из состояния (x, y) соответствуют переходам из состояния (x', y'), где $x' = x – 3$, $y' = y – 3$.

Необходимо минимизировать число потерянных элементов при переполнении очередей. Иначе говоря, нужно найти такое s, чтобы математическое ожидание доли времени, которое процесс проводит на отражающих экранах, было минимальным. Таким образом, в случае двух очередей задача оптимального разбиения общей памяти для FIFO-очередей может быть рассмотрена как одномерный случай задачи целочисленного программирования, где функция критерия оптимальности задается алгоритмически. Можно еще дать и такую интерпретацию решаемой задачи. Так как для каждого значения s мы имеем свою цепь Маркова, то можно сказать, что мы решаем задачу нахождения оптимальной цепи Маркова в смысле указанного критерия оптимальности. Для решения данной задачи используются результаты теории регулярных цепей Маркова.

Матрица переходных вероятностей

Представим случайное блуждание в виде регулярной марковской цепи. Пусть **P** — матрица переходов, $n = (m – s + 3)(s + 3)$ — количество состояний.

Определим нумерацию состояний, как показано на рис. 3. Сначала мы нумеруем состояния в области $x \le 0 \le s + 2$, $y \le 0 \le m – s + 2$. Затем состояния, в которых очереди переполняются, — это первый и второй отражающие экраны.



■ ***Рис. 3.*** Нумерация состояний при $m = 8$, $s = 4$

Утверждение. Пронумерованная таким образом матрица **P** имеет определенную структуру:

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \mathbf{Q}_3 \\ \mathbf{Q}_4 & \mathbf{Q}_5 & \mathbf{Q}_6 \\ & \mathbf{Q}_7 & \end{pmatrix},$$

где:

1) подматрица $\mathbf{Q}_1$ описывает блуждание в области $x \le s$, $y \le m - s$;

2) подматрицы $\mathbf{Q}_2$ и $\mathbf{Q}_3$ описывают переходы из области $x \le s$, $y \le m - s$ на первый и второй отражающие экраны;

3) подматрицы $\mathbf{Q}_4$, $\mathbf{Q}_5$ и $\mathbf{Q}_6$ описывают поведение процесса, когда очереди переполнены;

4) подматрица $\mathbf{Q}_7$ описывает переходы со второго отражающего экрана.

Лемма. Матрица $\mathbf{Q}_1$ размера $(s + 1)(m - s + 1)$ имеет следующий вид:

$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{D} & \mathbf{A} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{B} & \mathbf{C} & \mathbf{A} & \mathbf{O} & \cdots \\ \mathbf{O} & \mathbf{B} & \mathbf{C} & \ddots & \cdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{A} \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{B} & \mathbf{F} \end{pmatrix},$$

где **A**, **B**, **C**, **D**, **F** являются подматрицами размера $(s + 1)$; **O** — нулевая матрица;

$$\mathbf{A} = \begin{pmatrix} p_1r_2 + p_1q_2 + p_{12}q_2 & p_{12}r_2 & 0 & \cdots & 0 \\ p_1q_2 & p_1r_2 & p_{12}r_2 & \cdots & 0 \\ 0 & p_1q_2 & p_1r_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & p_{12}r_2 \\ 0 & 0 & \cdots & p_1q_2 & p_1r_2 \end{pmatrix}.$$

Мы докажем лемму с помощью математической индукции.

Доказательство:

1. База индукции. Пусть общий объем памяти равен $m = 2$; $s = 1$ — количество памяти, выделенное первой очереди. Тогда размер матрицы $\mathbf{Q}_1$ будет 4×4. Подматрицы **A**, **B**, **C**, **D**, **F** имеют размер 2×2 (подматрица **C** здесь не представлена ввиду маленького значения памяти m). Матрица $\mathbf{Q}_1$ имеет следующую форму:

$$\mathbf{Q}_1 = \left(\begin{array}{cc:cc} p_1q_1 + p_2q_2 + p_1q_{12} + \\ + p_2q_{12} + p_{12}q_{12} + r_1q_1 + \\ + r_1q_2 + r_1q_{12} + r_1r_2 & p_2r_2 + p_2q_1 + p_{12}q_1 & p_1r_2 + p_1q_2 + p_{12}q_2 & p_{12}r_2 \\ r_1q_2 + r_1q_{12} + p_1q_{12} & p_1q_1 + r_1q_1 + r_1r_2 & p_1q_2 & p_1r_2 \\ \hdashline r_1q_1 + r_1q_{12} + p_2q_{12} & p_2q_1 & p_2q_2 + r_1q_1 + r_1r_2 & p_2r_2 \\ r_1q_{12} & r_1q_1 & r_1q_2 & r_1r_2 \end{array}\right).$$

2. Индуктивное предположение. Предположим, что для размера памяти m лемма верна. Размерность $\mathbf{Q}_1$ будет $(s + 1)(m - s + 1)$, размерность подматриц — $(s + 1)$.

3. Индуктивный переход. Проверим, что при размере памяти $(m + 1)$ лемма верна. Так как добавилась еще одна единица памяти, то увеличилась одна из очередей. Рассмотрим два случая:

а. Единица памяти попала в первую очередь, размеры памяти m и s увеличились на единицу: $m = m + 1$, $s = s + 1$. Тогда область случайных блужданий увеличится по оси $0x$ — будет добавлено $(m - s + 1)$ новых состояний. Размерность матрицы $\mathbf{Q}_1$ увеличится на $(m - s + 1)$, размерность всех подматриц увеличится на единицу, и их количество останется прежним.

Пусть размерность подматриц будет $(s + 2)$. Проследим за изменением структуры на примере подматрицы **A**. К ней будут добавлены одна колонка и один столбец, и ее вид будет следующим:

$$\mathbf{A}_{(s+2)\times(s+2)} = \left(\begin{array}{ccc:c} & \mathbf{A}_{(s+1)\times(s+1)} & & 0 \\ & & & \vdots \\ & & & p_{12}r_2 \\ \hdashline 0 & \cdots & p_1q_2 & p_1r_2 \end{array}\right).$$

Общий вид подматрицы не изменится. Аналогично это можно показать и для остальных подматриц.

б. Единица памяти была добавлена во вторую очередь. Размеры m и s станут $m = m + 1$, $s = s$.

Область блуждания поднимется вверх по оси $0y$, т. е. будет добавлено $(s + 1)$ новых состояний. Размерность матрицы $\mathbf{Q}_1$ увеличится на $(s + 1)$. Размерности подматриц не изменятся, но количество подматриц **A**, **B**, **C** увеличится на один, т. е. общий вид матрицы $\mathbf{Q}_1$ сохранится.

Таким образом, в обоих случаях матрица $\mathbf{Q}_1$ не изменяет свою структуру — лемма доказана.

Используя этот метод, можно доказать, что и остальные подматрицы **P** сохраняют свой вид. Утверждение доказано.

Некоторые примеры численного анализа

Разработан алгоритм и программа нахождения оптимального разбиения памяти между очередями в зависимости от вероятностных характеристик очередей. Для решения поставленной задачи использовался аппарат управляемых случайных блужданий, регулярных цепей Маркова, система Intel® Math Kernel Library PARDISO. Вычисления производились с помощью кластера КарНЦ РАН.

■ Сравнение потерь

Входные данные	Величина потерь при переполнении ($m = 10$)	
	Оптимальное разбиение	Разбиение пополам ($s = 5$)
$p_1 = 0{,}25$, $p_2 = 0{,}25$, $p_{12} = 0{,}25$, $r_1 = 0{,}25$, $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,089 ($s = 5$)	0,089
$p_1 = 0{,}30$, $p_2 = 0{,}20$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$, $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,012 ($s = 6$)	0,014
$p_1 = 0{,}30$, $p_2 = 0{,}20$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$, $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,019 ($s = 7$)	0,025
$p_1 = 0{,}30$, $p_2 = 0{,}20$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$, $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,035 ($s = 7$)	0,046
$p_1 = 0{,}30$, $p_2 = 0{,}20$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$, $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,064 ($s = 8$)	0,075

Некоторые результаты вычислений приведены в таблице (указанные результаты были подтверждены имитационными экспериментами). Так как аналитическое решение не было получено, нам нужно решать задачу для отдельных m. $m = 10$ используется для примера. Взятые здесь вероятности являются теоретическими — для большей наглядности результатов. На практике же эти вероятности должны быть получены в результате предварительных статистических исследований.

Заключение

Анализируя результаты, можно сказать, что с увеличением вероятности включения в одну очередь (в данном случае — в первую) и уменьшением вероятности включения в другую при разбиении памяти пополам потери при переполнении увеличиваются. При оптимальном разбиении мы выделяем одной очереди s единиц памяти, а второй — $m–s$ единиц памяти. Так, при вероятностях включения 0,35 в одну очередь и 0,15 в другую разница потерь между оптимальным разбиением и разбиением пополам составляет 0,006. То есть из тысячи пакетов мы теряем на 6 пакетов меньше, разбивая память оптимально. А при вероятностях включения 0,45 и 0,05 соответственно — уже на 11 пакетов меньше.

Остается открытым вопрос о том, какой из методов работы с очередями (описанный здесь принцип четных–нечетных шагов или такой, где операции совершаются на любом шаге) будет оптимальным для тех или иных аппаратных или программных решений. В любом случае математические модели таких процессов должны функционировать в дискретном времени. Таким образом, использование классического аппарата теории массового обслуживания (например, основанного на пуассоновском потоке заявок с непрерывным временем) в качестве математических моделей таких систем не представляется оправданным.

Работа выполнена при финансовой поддержке РФФИ, грант 15-01-03404-а.

Литература

1. **Knuth D.** The Art of Computer Programming. — Addison-Wesley Professional, 1997. Vol. 1. — 672 p.
2. **Sedgewick R.** Algorithms in C++. Parts 1–4. — Addison-Wesley Professional, 1998. — 752 p.
3. **Bollapragada V., Murphy C., White R.** Inside Cisco IOS Software Architecture. — Cisco Press, 2000. — 240 p.
4. **Калачев А. В.** Многоядерные процессоры. — М.: БИНОМ, 2014. — 247 с.
5. **Yao A. C.** An Analysis of a Memory Allocation Scheme for Implementing Stacks // SIAM Journal on Computing. 1981. Vol. 10. P. 398–403.
6. **Flajolet P.** The Evolution of Two Stacks in Bounded Space and Random Walks in a Triangle // Lecture Notes in Computer Science. 1986. Vol. 223. P. 325–340.
7. **Louchard G., Schott R.** Probabilistic Analysis of Some Distributed Algorithms // Lecture Notes in Computer Science. 1990. Vol. 431. P. 239–253.

8. **Louchard G., Schott R., Tolley M., Zimmermann P.** Random Walks, Heat Equation and Distributed Algorithms // Journal of Computational and Applied Mathematics. 1994. N 53. P. 243–274.
9. **Maier R. S.** Colliding Stacks: A Large Deviations Analysis // Random Structures and Algorithms. 1991. N 2. P. 379–421.
10. **Соколов А. В.** О распределении памяти для двух стеков // Автоматизация эксперимента и обработки данных: сб. Петрозаводск: Карельский филиал АН СССР, 1980. С. 65–71.
11. **Aksenova E. A., Sokolov A. V.** The Optimal Implementation of Two FIFO-Queues in Single-Level Memory // Applied Mathematics. 2011. Vol. 2. P. 1297–1302.
12. **Sokolov A. V., Drac A. V.** The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory // Information Processing Letters. 2013. Vol. 13. P. 832–835.
13. **Аксенова Е. А., Драц А. В., Соколов А. В.** Оптимальное управление *n* FIFO-очередями на бесконечном времени // Информационно-управляющие системы. 2009. № 6. С. 46–54.
14. **Драц А. В., Соколов А. В.** Математический анализ процесса работы с M FIFO-очередями // Стохастическая оптимизация в информатике. 2012. Т. 8. № 2. С. 75–82.
15. **Kemeny J. G., Snell J. L.** Finite Markov Chains. — Springer, 1983. — 226 p.

UDC 004.942
doi:10.15217/issn1684-8853.2015.5.65

Optimal Control of Two Parallel FIFO Queues on an Infinite Time

Barkovsky E. A.[a], Post-Graduate Student, barkevgen@gmail.com
Sokolov A. V.[a, b], Dr. Sc., Phys.-Math., Professor, avs@krc.karelia.ru
[a]Institute of Applied Mathematical Research of Karelian Research Centre Russian Academy of Sciences, 11, Pushkinskaya St., 185910, Petrozavodsk, Russian Federation
[b]Petrozavodsk State University, 33, Lenin St., 185910, Petrozavodsk, Russian Federation

Introduction: FIFO queue is a very common data structure used in many hardware and software applications. In the development of various network devices and embedded operating systems, you often have to work with several FIFO queues located in a shared memory space. Also, there are architectures of multicore processors where two FIFO queues are allocated to one core. The purpose of this research is constructing and analyzing a mathematical model of two serial cyclic FIFO queues in a shared memory with the following protocol: insertion into one of the queues is performed on an odd step, and deletion is performed on an even step (the execution of the operations can be either serial or parallel). **Results:** The problem of optimal partitioning of the shared memory for FIFO-queues is formulated as a problem of integer programming, where the optimality criterion function is defined algorithmically. Mathematical and simulation models of this process were constructed for two queues, and numerical experiments based on theoretical data were performed. The mathematical model was built as a random walk on a two-dimensional integer lattice with reflecting screens, i.e. we deal with a regular uniform Markov chain. The optimality criterion is the minimum average share of lost (due to overflow) elements of the queues. The peculiarity of our approach is a specific execution of the queue operations: insertion and deletion of the elements occur according to the step (some amendments have been made to maintain homogeneity and regularity of the chain), and parallel execution of the operations is also possible. **Practical relevance:** With the help of the developed model, you can find the optimal partition of a limited shared memory to improve the system stability. The proposed models, algorithms and the developed software complex can be used in the design of networking devices, for example, routers, where packet losses are acceptable but unwanted. When we partition a shared memory for queues in the optimal way, we lose fewer packets and, as a result, the data is delivered quicker.

Keywords — Data Structures, FIFO queues, Random Walks, Regular Markov Chains.

References

1. Knuth D. *The Art of Computer Programming*. Addison-Wesley Professional, 1997. Vol. 1. 672 p.
2. Sedgewick R. *Algorithms in C++*. Parts 1–4. Addison-Wesley Professional, 1998. 752 p.
3. Bollapragada V., Murphy C., White R. *Inside Cisco IOS Software Architecture*. Cisco Press, 2000. 240 p.
4. Kalachev A. V. *Mnogoiadernye protsessory* [Multicore Architectures]. Moscow, BINOM, 2014. 247 p. (In Russian).
5. Yao A. C. An Analysis of a Memory Allocation Scheme for Implementing Stacks. *SIAM Journal on Computing*, 1981, vol. 10, pp. 398–403.
6. Flajolet P. The Evolution of Two Stacks in Bounded Space and Random Walks in a Triangle. *Lecture Notes in Computer Science*, 1986, vol. 223, pp. 325–340.
7. Louchard G., Schott R. Probabilistic Analysis of Some Distributed Algorithms. *Lecture Notes in Computer Science*, 1990, vol. 431, pp. 239–253.
8. Louchard G., Schott R., Tolley M., Zimmermann P. Random Walks, Heat Equation and Distributed Algorithms. *Journal of Computational and Applied Mathematics*, 1994, no. 53, pp. 243–274.
9. Maier R. S. Colliding Stacks: A Large Deviations Analysis. *Random Structures and Algorithms*, 1991, no. 2, pp. 379–421.
10. Sokolov A. V. About Memory Distribution for Two Stacks. *Avtomatizatsiia eksperimenta i obrabotki dannykh*, Petrozavodsk, Karel'skii filial AN SSSR, 1980, pp. 65–71 (In Russian).
11. Aksenova E. A., Sokolov A. V. The Optimal Implementation of Two FIFO-Queues in Single-Level Memory. *Applied Mathematics*, 2011, vol. 2, pp. 1297–1302.
12. Sokolov A. V., Drac A. V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory. *Information Processing Letters*, 2013, vol. 13, pp. 832–835.
13. Aksenova E. A., Drac A. V., Sokolov A. V. Optimal Control of the *n* FIFO-queues for Infinity Time. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2009, no. 6, pp. 46–54 (In Russian).
14. Drac A. V., Sokolov A. V. Mathematical Analysis of Processing M FIFO-queues. *Stokhasticheskaia optimizatsiia v informatike*, 2012, vol. 8, no. 2, pp. 75–82 (In Russian).
15. Kemeny J. G., Snell J. L. *Finite Markov Chains*. Springer, 1983. 226 p.