УДК 004.942
doi:10.15217/issn1684-8853.2016.1.65

МОДЕЛЬ УПРАВЛЕНИЯ ДВУМЯ ПАРАЛЛЕЛЬНЫМИ FIFO-ОЧЕРЕДЯМИ, ДВИГАЮЩИМИСЯ ДРУГ ЗА ДРУГОМ В ОБЩЕЙ ПАМЯТИ

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

Постановка проблемы: *при разработке многих аппаратных и программных приложений применяют структуру данных «FIFO-очередь». В различных сетевых устройствах и встроенных операционных системах применяется несколько FIFO-очередей, расположенных в общем пространстве памяти. Также существуют архитектуры многоядерных процессоров, где каждому ядру выделено две FIFO-очереди. Программные и аппаратные решения, применяемые для реализации описанных задач, должны увеличивать надежность (снижение доли потерянных при переполнении памяти элементов очередей) работы таких устройств.* ***Цель:*** *построение и анализ математической модели процесса работы с двумя FIFO-очередями, двигающимися друг за другом по кругу, в общей памяти, где 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:*** *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 целочисленной пирамиде с отражающими экранами. Предложен алгоритм нумерации состояний, установлен вид матрицы переходных состояний полученной регулярной цепи Маркова (доказаны соответствующие утверждения), разработаны алгоритм и программа вычисления средней доли потерянных при переполнении элементов очередей. Особенностью данного исследования является специфическое выполнение операций над очередями: включение и исключение элементов происходит в зависимости от шага (были сделаны поправки для сохранения качеств однородности и регулярности цепи) и создана возможность выполнять операции параллельно.* ***Практическая значимость:*** *предложенные модель, алгоритм и программный комплекс для анализа метода движения очередей «друг за другом» могут применяться при проектировании сетевых устройств, например маршрутизаторов, микросхем, реализующих работу с несколькими FIFO-очередями, и других программных и аппаратных устройств, где потери элементов являются допустимой, но нежелательной ситуацией. С помощью разработанной модели можно, при заданных вероятностных характеристиках очередей, выбрать лучший метод представления очередей, например, из двух методов — классического последовательного циклического метода или метода «Друг за другом».*

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

Введение

В большом количестве приложений используются несколько FIFO-очередей, расположенных в общем пространстве памяти. При разработке сетевых устройств и встроенных операционных систем требуются эффективные алгоритмы работы с этими структурами данных — в приложениях управляющих потоками пакетов Internet, требования на время обработки пакетов маршрутизатором могут быть очень жесткие. Так, в Cisco IOS механизм страничной виртуальной памяти не используется, вместо него имеются пулы оперативной памяти, где и происходит вся работа. Для представления FIFO-очередей применяют различные программные или аппаратные решения [1–3].

Нужно также отметить, что существуют архитектуры многоядерных процессоров без кэш-памяти. Например, в архитектуре AsAP-II каждое ядро имеет два FIFO-буфера, а в архитектуре SEAforth — два стека [4]. Очереди и стеки реализованы циклически и раздельно, при переполнении происходит потеря элементов. В наших задачах для хранения нескольких структур данных используется общая память. В ряде случаев это позволяет снизить потери элементов при переполнении.

Модель последовательного, связанного и страничного способов представления нескольких FIFO-очередей в памяти одного уровня описаны в работах [5–8]. Здесь предполагается, что на каждом шаге дискретного времени происходят некоторые операции со структурами данных (с заданными вероятностями). Так как время выполнения операций — не случайная величина, а константа, фиксированным является и шаг времени. Первоначально такие модели в виде случайного блуждания в треугольнике [9–14] были построены для решения задачи анализа процесса работы с двумя стеками, растущими навстречу друг другу, поставленной в работе [3].

Немного другой способ выполнения операций с несколькими FIFO-очередями в общей памяти предложен в работе [1]. В этом способе на нечетном шаге допускаются только операции включения элементов в одну из n очередей, а на четном

шаге — исключения из очередей (в первом и втором случае операции равновероятны). Там была поставлена задача разработки математической модели для описания работы системы таких очередей. При этом не рассматривался конкретный способ представления очередей в памяти, т. е. предполагалось, что очереди могут быть неограниченной длины, что на практике невыполнимо.

Авторами [15] предложена математическая модель и решена задача оптимального разбиения общей памяти для двух FIFO-очередей в случае их последовательного циклического представления. Операции с очередями (с заданными вероятностями) выполнялись по вышеупомянутому принципу как последовательно, так и параллельно. В качестве критерия оптимальности рассмотрена минимальная средняя доля потерянных при переполнении элементов на бесконечном времени работы.

Этот критерий оптимальности стоит рассматривать, если переполнение очереди является стандартной ситуацией. Когда размер очереди превышает размер предоставленной ей памяти, все поступающие в нее элементы отбрасываются до тех пор, пока не появится свободный участок памяти (этот участок появится только после исключения элемента из очереди). Сетевые маршрутизаторы [2] работают по этой схеме. Такие потери элементов приводят к нежелательному результату. Например, в некоторых сетевых протоколах отброшенные пакеты будут посылаться снова, что приводит к замедлению работы системы — ее эффективность падает. Поэтому число таких ситуаций необходимо свести к минимуму.

В данной работе мы предлагаем математическую и имитационную модели процесса работы с двумя параллельными очередями, когда они двигаются по кругу друг за другом. Здесь общая память заранее не делится между очередями. Этот метод работы с FIFO-очередями предложен в работе [16]. Математическая модель строится в виде случайного блуждания по целочисленной пирамиде. Предварительные результаты исследования докладывались на конференции [17].

Математическая модель

Пусть в памяти размера m единиц мы работаем с двумя циклическими параллельными FIFO-очередями, которые двигаются друг за другом по кругу. Возобновление движения пустой очереди начинается с середины промежутка между очередями.

Операции, производимые с очередями, выполняются по следующей схеме: на нечетном шаге совершается операция включения в одну из очередей (включение происходит в конец очереди), на четном шаге — операция исключения из какой-либо очереди (исключение происходит из начала очереди), причем известны некоторые вероятностные характеристики операций, производимых с очередями. Можно сказать, что у нас имеется регулярный поток событий, которые происходят через фиксированный шаг дискретного времени. Следует заметить, что этот поток не является простейшим (в терминологии теории массового обслуживания), так как у него есть последействие.

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 не будет являться регулярной и однородной, два последовательных шага объединяем в один, а также вводим следующие вероятности операций, не изменяющих длины очередей (например, чтение): r_o — на нечетном шаге и r_e — на четном шаге, при этом $r_o \neq 0$, $r_e \neq 0$. Соответственно: $p_1 + p_2 + r_o = 1$, $q_1 + q_2 + r_e = 1$.

Тогда состояние на каждом шаге определяется наступлением одной из следующих комбинаций событий, где сумма всех вероятностей равна 1:

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

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

— включение в первую очередь с вероятностью $p_1 r_e + p_{12} q_2$;

— включение во вторую очередь с вероятностью $p_2 r_e + p_{12} q_1$;

— включение параллельно в обе очереди с вероятностью $p_{12} r_e$;

— исключение из первой очереди с вероятностью $q_1 r_o + p_2 q_{12}$;

— исключение из второй очереди с вероятностью $q_2 r_o + p_1 q_{12}$;

— исключение параллельно из обеих очередей с вероятностью $q_{12} r_o$;

— выполнение над очередями сохраняющих их состояние противоположных операций с вероятностью $r_o r_e + p_1 q_1 + p_2 q_2 + p_{12} q_{12}$.

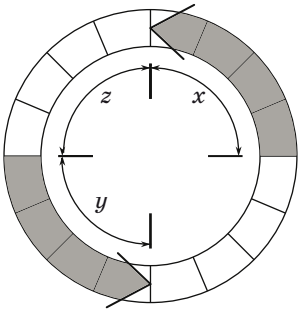
Предполагается, что в очередях хранятся данные фиксированного размера. При исключении информации из пустой очереди не происходит завершения работы.

Целью исследования является определение средней доли потерянных элементов для сравнения со средней долей потерянных элементов при последовательном циклическом способе организации параллельных очередей в случае оптимального разбиения общей памяти [15]. По закону больших чисел для регулярных цепей Маркова

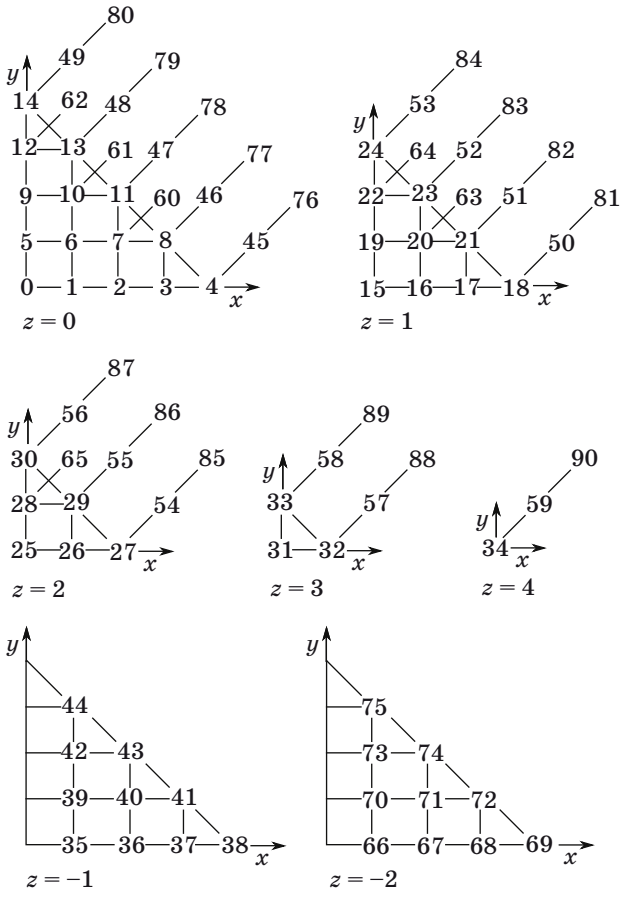
[18] это эквивалентно нахождению решения, вычисляющего значение вероятности переполнения памяти на бесконечном промежутке времени.

Обозначим через x и y текущие длины очередей, через z — расстояние между концом первой очереди и началом второй. Схема движения очередей показана на рис. 1.

В качестве математической модели рассмотрим случайное блуждание (которое конкретизировано с учетом специфики объекта исследования) по целочисленной трехмерной пирамиде с вершиной (0;0;0) и основанием $x + y + z = m$ (рис. 2).



■ ***Рис. 1.*** Схема движения очередей



■ ***Рис. 2.*** Область блуждания, нумерация состояний при $m = 4$

Для вычисления средней доли потерянных при переполнении элементов очередей введены искусственные отражающие экраны:

1) $z = -1$ и $x + y' + z = m + 1$ ($y' = y + 1$). Блуждание переходит на экран $z = -1$, если происходит включение в первую очередь в то время, когда $z = 0$, и на экран $x + y' + z = m + 1$, если происходит включение во вторую очередь в то время, когда $x + y + z = m$. В данном случае элемент будет потерян, а очереди будут находиться на экранах до тех пор, пока поступают новые элементы. В случае исключения элемента очереди вернутся в состояния, принадлежащие пирамиде;

2) в случае когда происходит включение во вторую очередь и одновременное исключение из первой, в то время когда $x + y + z = m$, происходит потеря элемента и увеличение расстояния между концом второй очереди и началом первой. Для такой ситуации также необходимо ввести отражающий экран — $x' = x - 1$;

3) если происходит операция одновременного включения и исключения элемента из первой очереди при $z = 0$ либо операция включения и исключения из второй очереди при $x + y + z = m$, происходит потеря элемента, и очереди возвращаются в состояния, принадлежащие пирамиде. Для учета потери элемента вводим экраны $z = -2$ и $x + y' + z = m + 2$ ($y' = y + 2$) соответственно.

Определим схему переходов между состояниями. Величина *distance* означает расстояние между концом первой и началом второй очереди. Если оживает первая очередь, то $\text{distance} = \left[\frac{m-y-1}{2}\right]$, и $\text{distance} = \left[\frac{m-x-1}{2}\right]$, если оживает вторая очередь. Пусть (x, y, z) — текущее состояние процесса, тогда блуждание по пирамиде можно описать следующим образом:

$$(x, y, z) \xrightarrow[r_o r_e]{} (x, y, z);$$

$$(x, y, z) \xrightarrow[p_1 r_e]{} (x', y', z') =$$

$$= \begin{cases} (x+1, y, z-1) & x, y, z > 0 \\ (x+1, 0, 0) & y = 0, x < m \\ (1, y, \text{distance} - 1) & x = 0, y < m \\ (x, y, -1) & z = 0, y \neq 0 \text{ либо } x = m \end{cases};$$

$$(x, y, z) \xrightarrow[p_2 r_e]{} (x', y', z') =$$

$$= \begin{cases} (x, y+1, z) & x, y, z > 0 \\ (x, 1, \text{distance}) & y = 0, x < m \\ (0, y, 0) & x = 0, y < m \\ (x, y+1, z) & x + y + z = m \end{cases};$$

$$(x,y,z)\xrightarrow[r_oq_1]{}(x',y',z')=$$
$$=\begin{cases}(0,y,0) & x=0 \text{ либо } x=1\\ (x-1,0,0) & y=0\\ (x-1,y,z) & \text{иначе}\end{cases};$$

$$(x,y,z)\xrightarrow[r_oq_2]{}(x',y',z')=$$
$$=\begin{cases}(x,0,0) & y=0 \text{ либо } y=1\\ (0,y-1,0) & x=0\\ (x,y-1,z+1) & \text{иначе}\end{cases};$$

$$(x,y,z)\xrightarrow[p_1q_2]{}(x',y',z')=$$
$$=\begin{cases}(x+1,y-1,z) & x,y,z>0\\ (x+1,0,0) & x<m, y=0 \text{ либо } y=1\\ (1,y-1,\text{distance}) & x=0, y<m\\ (x,y-1,-1) & z=0, y\neq 0 \text{ либо } x=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_2q_1]{}(x',y',z')=$$
$$=\begin{cases}(x-1,y+1,z) & x,y,z>0\\ (x-1,1,\text{distance}) & y=0, x<m\\ (0,y+1,0) & y<m, x=0 \text{ либо } x=1\\ (x',y,z) & x+y+z=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_1q_1]{}(x',y',z')=$$
$$=\begin{cases}(x,y,z-1) & x,y,z>0\\ (0,y,0) & x=0\\ (x,0,0) & y=0\\ (x,y,-2) & z=0, y\neq 0 \text{ либо } x=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_2q_2]{}(x',y',z')=$$
$$=\begin{cases}(x,y,z+1) & x,y,z>0\\ (x,0,0) & y=0\\ (0,y,0) & x=0\\ (x,y+2,z+1) & x+y+z=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_{12}r_e]{}(x',y',z')=$$
$$=\begin{cases}(x+1,y+1,z-1) & x,y,z>0\\ (x+1,1,\text{distance}-1) & y=0, x<m\\ (1,y+1,\text{distance}-1) & x=0, y<m\\ (x,y+1,-1) & z=0, y\neq 0 \text{ либо } x=m\\ (x+1,y+1,z-1) & x+y+z=m\end{cases};$$

$$(x,y,z)\xrightarrow[r_oq_{12}]{}(x',y',z')=$$
$$=\begin{cases}(0,y-1,0) & x=0 \text{ либо } x=1\\ (x-1,0,0) & y=0 \text{ либо } y=1\\ (x-1,y-1,z+1) & \text{иначе}\end{cases};$$

$$(x,y,z)\xrightarrow[p_1q_{12}]{}(x',y',z')=$$
$$=\begin{cases}(x,y-1,z) & x,y,z>0\\ (0,y-1,0) & x=0\\ (x-1,0,0) & y=0\\ (x,y-1,-2) & z=0, y\neq 0 \text{ либо } x=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_2q_{12}]{}(x',y',z')=$$
$$=\begin{cases}(x-1,y,z+1) & x,y,z>0\\ (x-1,0,0) & y=0\\ (0,y-1,0) & x=0\\ (x-1,y+2,z+1) & x+y+z=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_{12}q_1]{}(x',y',z')=$$
$$=\begin{cases}(x,y+1,z-1) & x,y,z>0\\ (x,1,\text{distance}-1) & y=0, x<m\\ (0,y+1,0) & x=0, y<m\\ (x,y+1,-2) & z=0, y\neq 0 \text{ либо } x=m\\ (x',y,z-1) & x+y+z=m\end{cases};$$

$$(x\,y,z)\xrightarrow[p_{12}q_2]{}(x',y',z')=$$
$$=\begin{cases}(x+1,y,z) & x,y,z>0\\ (x+1,0,0) & y=0, x<m\\ (1,y,\text{distance}) & x=0, y<m\\ (x,y,-1) & z=0, y\neq 0 \text{ либо } x=m\\ (x+1,y+2,z) & x+y+z=m\end{cases};$$

$$(x,y,z)\xrightarrow[p_{12}q_{12}]{}(x',y',z')=$$
$$=\begin{cases}(x,y,z) & x,y,z>0\\ (0,y,0) & x=0\\ (x,0,0) & y=0\\ (x,y,-2) & z=0, y\neq 0 \text{ либо } x=m\\ (x,y+2,z) & x+y+z=m\end{cases}.$$

Переходы с *первого* фиктивного экрана:
1) переходы с плоскости $z=-1$:

$$(x,y,-1)\xrightarrow[r_or_e]{}(x,y,0);$$

$$(x,y,-1)\xrightarrow[p_1r_e]{}(x,y,-1);$$

$$(x,y,-1)\xrightarrow[p_2r_e]{}(x,y+1,0);$$

$$(x,y,-1)\xrightarrow[r_oq_1]{}(x-1,y,0);$$
$$(x,y,-1)\xrightarrow[r_oq_2]{}(x',y',z')=$$
$$=\begin{cases}(x,0,0) & y=0 \text{ либо } y=1\\ (x,y-1,1) & y>1\end{cases};$$

$$(x, y, -1) \xrightarrow{p_1 q_1} (x, y, -2);$$

$$(x, y, -1) \xrightarrow{p_2 q_2} (x', y', z') = \begin{cases} (x, y+2, 1) & x + y = m \\ (x, y, 1) & \text{иначе} \end{cases};$$

$$(x, y, -1) \xrightarrow{p_1 q_2} (x', y', z') = \begin{cases} (x, 0, -1) & y = 0 \\ (x, y-1, -1) & \text{иначе} \end{cases};$$

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

$$(x, y, -1) \xrightarrow{p_{12} r_e} (x, y+1, -1);$$

$$(x, y, -1) \xrightarrow{r_o q_{12}} (x', y', z') = \begin{cases} (x-1, 0, 0) & y = 0 \text{ либо } y = 1 \\ (x-1, y-1, 1) & y > 1 \end{cases};$$

$$(x, y, -1) \xrightarrow{p_1 q_{12}} (x, y-1, -2);$$

$$(x, y, -1) \xrightarrow{p_2 q_{12}} (x', y', z') = \begin{cases} (x-1, y+2, 1) & x + y = m \\ (x-1, y, 1) & \text{иначе} \end{cases};$$

$$(x, y, -1) \xrightarrow{p_{12} q_1} (x', y', z') = \begin{cases} (x, y, -2) & x + y = m \\ (x, y+1, -2) & \text{иначе} \end{cases};$$

$$(x, y, -1) \xrightarrow{p_{12} q_2} (x', y', z') = \begin{cases} (x, y+2, 1) & x + y = m \\ (x, y, -1) & \text{иначе} \end{cases};$$

$$(x, y, -1) \xrightarrow{p_{12} q_{12}} (x, y, -2);$$

2) переходы из состояния $x + y' + z = m + 1$ $(y' = y + 1)$:

$$(x, y+1, z) \xrightarrow{r_o r_e} (x, y, 0);$$

$$(x, y+1, z) \xrightarrow{p_1 r_e} (x', y', z') = \begin{cases} (0, y+1, 0) & y = m \\ (x+1, y, z-1) & \text{иначе} \end{cases};$$

$$(x, y+1, z) \xrightarrow{p_2 r_e} (x, y+1, z);$$

$$(x, y+1, z) \xrightarrow{r_o q_1} (x', y', z') = \begin{cases} (0, y, 0) & x = 0 \text{ либо } x = 1 \\ (x-1, y, z) & \text{иначе} \end{cases};$$

$$(x, y+1, z) \xrightarrow{r_o q_2} (x', y', z') = \begin{cases} (0, y-1, 0) & x = 0 \\ (x, y-1, z+1) & \text{иначе} \end{cases};$$

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

$$(x, y+1, z) \xrightarrow{p_2 q_1} (x', y', z') = \begin{cases} (0, y+1, 0) & x = 0 \\ (x', y, z) & x \neq 0, y \neq 0 \end{cases};$$

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

$$(x, y+1, z) \xrightarrow{p_2 q_2} (x, y+2, z);$$

$$(x, y+1, z) \xrightarrow{p_{12} r_e} (x', y', z') = \begin{cases} (0, y+1, 0) & y = m \\ (x+1, y+1, z-1) & \text{иначе} \end{cases};$$

$$(x, y+1, z) \xrightarrow{r_o q_{12}} (x', y', z') = \begin{cases} (0, y-1, 0) & x = 0 \text{ либо } x = 1 \\ (x-1, y-1, z+1) & \text{иначе} \end{cases};$$

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

$$(x, y+1, z) \xrightarrow{p_2 q_{12}} (x-1, y+2, z);$$

$$(x, y+1, z) \xrightarrow{p_{12} q_1} (x', y', z') = \begin{cases} (0, y+1, 0) & x = 0 \\ (x', y, z-1) & x + y + z = m \end{cases};$$

$$(x, y+1, z) \xrightarrow{p_{12} q_2} (x', y', z') = \begin{cases} (0, y+2, 0) & y = m \\ (x+1, y+2, z-1) & \text{иначе} \end{cases};$$

$$(x, y+1, z) \xrightarrow{p_{12} q_{12}} (x', y', z') = \begin{cases} (0, y+2, z-1) & x, y, z > 0 \\ (x, y+2, z) & \text{иначе} \end{cases};$$

3) переходы из состояний (x', y, z) $(x' = x - 1)$ повторяют переходы из соответствующих состояний, принадлежащих пирамиде:

$$(x', y, z) \leftrightarrow \begin{cases} (0, y, 0) & x' = 0 \\ (x', y, z) & \text{иначе} \end{cases}.$$

Переходы из состояний *второго* фиктивного экрана повторяют переходы из соответствующих состояний, принадлежащих пирамиде:

$$(x, y, z) \leftrightarrow (x-1, y, z+2) \text{ для } z = -2;$$

$$(x, y', z) \leftrightarrow \begin{cases} (x, 0, 0) & y = 0 \\ (x-1, 0, 0) & y = 1 \\ (0, y-1, 0) & y + z = m \\ (x, y, z+1) & \text{иначе} \end{cases}$$

для $x + y' + z = m + 2$.

Случайное блуждание рассматриваем в виде регулярной конечной цепи Маркова с переходной матрицей **P**. Зададим нумерацию так, как показано на рис. 2: сначала нумеруем состояния, принадлежащие основанию пирамиды ($z = 0$), далее — поперечным сечениям ($z = 1, 2, \ldots, m - 1$) и высоте пирамиды ($z = m$), затем — состояния первого фиктивного экрана, последними нумеруем состояния второго экрана.

Нумерацию начинаем с нуля, общее количество состояний в цепи будет

$$n = \sum_{i=0}^{m} \frac{(i+1)(i+2)}{2} + \frac{5m^2 + 7m + 4}{2}.$$

Далее, согласно введенной нумерации состояний и вышеуказанной схеме переходов, составляется матрица переходных вероятностей **P**. В данном исследовании был установлен вид матрицы **P** для произвольных значений параметра m. При составлении матрицы для каждого конкретного состояния определяются те состояния, в которые процесс переходит при выполнении допустимых операций, и вычисляются соответствующие вероятности переходов. Данный процесс был автоматизирован с помощью программы на языке C.

Следующим шагом является решение уравнения $\boldsymbol{\alpha}\mathbf{P} = \boldsymbol{\alpha}$, где $\boldsymbol{\alpha}$ — предельный вектор для полученной марковской цепи. Для этого применялась система Intel Math Kernel Library PARDISO. Данный инструмент используется для прямого решения систем линейных алгебраических уравнений с разряженной матрицей.

Элемент вектора $\boldsymbol{\alpha}_i$ — это средняя доля времени, которое процесс проводит в состоянии i [18]. Для вычисления времени, проводимого процессом в состояниях, где происходят потери элементов очередей, нужно просуммировать элементы вектора $\boldsymbol{\alpha}$, соответствующие состояниям на экранах. При введенной нумерации это будут последние $\frac{5m^2 + 7m + 4}{2}$ элементов вектора.

Оценка сложности алгоритма:

$$O\left(\left(\sum_{i=0}^{m} \frac{(i+1)(i+2)}{2} + \frac{5m^2 + 7m + 4}{2}\right)^3\right).$$

Предложенная математическая модель анализируемого в статье нового способа работы с FIFO-очередями в общей памяти требует проверки ее адекватности. Для этого была разработана имитационная модель процесса, краткое изложение которой представлено в следующем разделе.

Имитационная модель

Обозначим через x текущую длину первой очереди, через y — второй, через z — расстояние между очередями. Общий объем памяти равен m. Тогда в качестве имитационной модели будем рассматривать случайное блуждание по кругу по часовой стрелке (см. рис. 1), при этом на нечетных шагах:

1) с вероятностью p_1 увеличивается длина первой очереди, соответственно, уменьшается расстояние z между очередями;

2) p_2 — увеличивается длина второй очереди;

3) p_{12} — одновременно увеличивается длина первой и второй очередей, расстояние z уменьшается;

4) r_o — процесс остается на месте.

На четных шагах:

1) с вероятностью q_1 уменьшается длина первой очереди;

2) q_2 — уменьшается длина второй очереди, соответственно, увеличивается расстояние z между очередями;

3) q_{12} — длина первой и второй очередей уменьшается одновременно, z увеличивается;

4) r_e — процесс остается на месте.

Для того чтобы в имитационной модели определять, какое действие будет выполняться, необходимо отрезок от 0 до 1 разделить между вероятностями так, чтобы их сумма была равна 1.

Далее с помощью датчика случайных чисел (использовался датчик, встроенный в транслятор gcc) генерируется последовательность, с помощью которой будет происходить блуждание по кругу, и подсчитывается количество нахождений очередей в переполнении, т. е. в случаях, когда $z = 0$ либо $x + y + z = m$.

Некоторые примеры численного анализа

Для определения случаев, когда целесообразно применять анализируемый в статье метод работы с очередями «Друг за другом», был проведен ряд экспериментов с различным набором исходных данных. Аналогичные вычисления были проведены для последовательного циклического способа организации очередей в общей памяти при оптимальном разбиении памяти [15]. Там же был рассмотрен случай, когда память заранее делится пополам при последовательном способе организации очередей. Это может быть полезно, если заранее не известны вероятностные

■ Сравнение потерь

Входные данные	Величина потерь при переполнении ($m = 10$)		
	Оптимальное разбиение	Разбиение пополам ($s = 5$)	Друг за другом
$p_1 = 0{,}25$, $p_2 = 0{,}25$, $p_{12} = 0{,}25$, $r_1 = 0{,}25$ $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,089 ($s = 5$)	0,089	0,100
$p_1 = 0{,}30$, $p_2 = 0{,}20$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$ $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,012 ($s = 6$)	0,014	0,019
$p_1 = 0{,}35$, $p_2 = 0{,}15$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$ $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,019 ($s = 7$)	0,025	0,024
$p_1 = 0{,}40$, $p_2 = 0{,}10$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$ $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,035 ($s = 7$)	0,046	0,034
$p_1 = 0{,}45$, $p_2 = 0{,}05$, $p_{12} = 0{,}10$, $r_1 = 0{,}40$ $q_1 = q_2 = q_{12} = r_2 = 0{,}25$	0,064 ($s = 8$)	0,075	0,059

характеристики операций, производимых с очередями.

Некоторые результаты вычислений (все указанные результаты были подтверждены имитационными экспериментами) представлены в таблице. Так как аналитическое решение не было получено, нам нужно проводить вычисления для конкретных значений m. Величину $m = 10$ мы используем в качестве примера, взятые здесь вероятности являются теоретическими — для большей наглядности результатов. На практике же эти вероятности должны быть получены в результате предварительных статистических исследований.

Анализируя результаты, можно сказать, что для случая равных вероятностей метод «Друг за другом» дает наибольшие потери. Но с увеличением вероятности включения в одну очередь (в данном случае — в первую) и уменьшением вероятности включения в другую этот метод дает наименьшие потери.

Так, при вероятностях включения 0,4 в одну очередь и 0,1 в другую разница потерь между «Друг за другом» и оптимальным разбиением — 0,001. То есть из тысячи элементов (пакетов, в случае сетевых приложений FIFO-очередей) мы теряем в среднем на один элемент меньше, пуская очереди друг за другом, по кругу. А при вероятностях включения 0,45 и 0,05 соответственно — уже на пять элементов меньше.

С помощью разработанного программного комплекса можно находить оптимальный способ представления очередей для их конкретных вероятностных характеристик.

Заключение

В данной статье мы предлагаем и анализируем математическую модель, которая описывает новый способ управления работой двумя параллельными FIFO-очередями в общей памяти. Такая модель может быть полезна при разработке различных технических систем, в архитектуре которых необходим именно такой способ управления.

И хотя в настоящее время память становится все больше и дешевле, есть устройства, где ее размер ограничен архитектурой, и имеющийся ресурс необходимо использовать эффективно. Это могут быть различные сетевые устройства (например, маршрутизаторы). В них нельзя беспредельно увеличивать размер памяти, так как при больших размерах очередей значительно увеличивается время обработки элементов. Сетевые протоколы устроены так, что при большом времени обработки пакеты считаются потерянными, и их посылают снова, тем самым увеличивая загрузку сети (congestion collapse) [19].

Другими такими устройствами могут быть микросхемы и микропроцессоры, которые используются в системах, где потеря элементов очередей является допустимой, но нежелательной ситуацией (например, цифровая обработка сигналов или работа с мультимедийными приложениями [4]). В таких процессорах каждая очередь может содержать всего несколько десятков элементов. Предложенный в статье способ может являться одним из методов повышения надежности (как снижения доли потерянных элементов) работы таких устройств.

Остается открытым вопрос, какой из методов работы (разделение на четные-нечетные шаги или произвольное выполнение операций) будет оптимальным для тех или иных аппаратных или программных решений. Отметим, что в реальной телекоммуникационной системе вероятностные характеристики очередей будут меняться со временем. При этом разумно предположить, что вероятности появления новых элементов в очередях будут периодически повторяться. Тогда интервал времени такого периода (например, сутки) можно разбить на участки, где эти вероятности мало меняются, и на каждом промежутке времени выбирать тот или иной метод представления очередей, например, друг за другом по кругу в общей памяти или когда каждая очередь представлена циклически отдельно в своем участке.

Отдельной задачей в случае метода движения друг за другом по кругу может быть задача нахождения оптимальной точки начала движения второй очереди или оптимального сдвига очереди в случае если одна очередь догнала другую.

Оценить масштабируемость предлагаемого решения достаточно сложно. Даже для трех очере-

дей возможны три варианта представления. Три очереди двигаются друг за другом в общей памяти; две очереди двигаются друг за другом в общей памяти, а третья представлена циклически в оставшейся памяти; или все три очереди представлены циклически, каждая в своем участке памяти. В случае n очередей число вариантов представления резко возрастает. Хотя, в принципе, можно предложить такой способ масштабирования решения: n очередей разбиваются на пары, и внутри каждой пары периодически производится смена метода представления, а в случае циклического раздельного представления очередей — оптимальное перераспределение памяти. Это делать проще, чем производить полное перераспределение памяти, но в таком случае достигается не глобальный, а некий локальный минимум средней доли потерянных при переполнении элементов очередей. Такой подход применяется в многоядерных процессорах.

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

Литература

1. **Sedgewick R.** Algorithms in C++. Parts 1–4. — Addison-Wesley Professional, 1998. — 752 p.
2. **Bollapragada V., Murphy C., White R.** Inside Cisco IOS Software Architecture. — Cisco Press, 2000. — 240 p.
3. **Knuth D.** The Art of Computer Programming. Vol. 1. — Addison-Wesley Professional, 1997. — 672 p.
4. **Калачев А. В.** Многоядерные процессоры. — М.: БИНОМ, 2014. — 247 с.
5. **Аксенова Е. А., Соколов А. В., Драц А. В.** Оптимальное управление n FIFO-очередями на бесконечном времени // Информационно-управляющие системы. 2009. № 6. С. 46–54.
6. **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.
7. **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.
8. **Соколов А. В., Драц А. В.** Моделирование некоторых методов представления n FIFO-очередей в памяти одного уровня // Эвристические алгоритмы и распределенные вычисления. 2014. Т. 1. № 1. С. 40–52.
9. **Соколов А. В.** О распределении памяти для двух стеков // Автоматизация эксперимента и обработки данных. 1980. С. 65–71.
10. **Yao A. C.** An Analysis of a Memory Allocation Scheme for Implementing Stacks // SIAM Journal on Computing. 1981. Vol. 10. P. 398–403.
11. **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.
12. **Louchard G., Schott R.** Probabilistic Analysis of Some Distributed Algorithms // Lecture Notes in Computer Science. 1990. Vol. 431. P. 239–253.
13. **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.
14. **Maier R. S.** Colliding Stacks: A Large Deviations Analysis // Random Structures and Algorithms. 1991. N 2. P. 379–421.
15. **Соколов А. В., Барковский Е. А.** Оптимальное управление двумя параллельными FIFO-очередями на бесконечном времени // Информационно-управляющие системы. 2015. № 5. С. 65–71. doi:10.15217/issn1684-8853.2015.5.65
16. **Соколов А. В.** Математические модели и алгоритмы оптимального управления динамическими структурами данных. — Петрозаводск: ПетрГУ, 2002. — 216 с.
17. **Sokolov A. V., Barkovsky E. A.** Some Problems of Optimal Control of Two Parallel FIFO-queues // Intern. Conf. on Numerical Analysis and Applied Mathematics: Proc. 12th Int. Conf., Rhodes, 18–22 Sept. 2014. AIP Publishing, 2015. Vol. 1648. P. 520003.
18. **Kemeny J. G., Snell J. L.** Finite Markov Chains. — Van Nostrand, 1969. — 210 p.
19. **Tanenbaum A. S., Wetherall D. J.** Computer Networks. — Pearson Education, 2011. — 933 p.

UDC 004.942
doi:10.15217/issn1684-8853.2016.1.65

Management Model for Two Parallel FIFO Queues Moving One after Another in Shared Memory

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

Introduction: FIFO queue is a popular data structure in hardware and software applications. Various network devices and embedded operating systems use several FIFO queues located in a shared memory space. There are also multi-core processor architectures where two FIFO queues are allocated for each core. Software and hardware applications used to solve the described problems should improve the device reliability, reducing the average portion of the queue elements lost by overflow. **Purpose:** The goal is to construct and analyze a mathematical model of operating two FIFO queues moving one after another in a circle in the shared memory. On an odd step, elements are inserted into one of the queues. On an even step, elements are deleted. Both serial and parallel execution of the operations are possible. **Results:** A mathematical and a simulation models of this process for two queues were constructed, and numerical experiments based on theoretical data were performed. The mathematical model is built as a random walk on an integer pyramid with reflecting screens. An algorithm was proposed for the numbering of states; the form of the matrix of transition states for the obtained regular Markov chain was established; and the corresponding propositions were proved. Another algorithm and software were developed for calculating the average portion of queue elements lost by overflow. The peculiarity of this research is a specific execution of operations with queues: insertion and deletion of elements occur depending on the step (some amendments were made to maintain the homogeneity and regularity of the chain). The operations can be executed in parallel. **Practical relevance:** The model, algorithm and software complex proposed for the analysis of "One after another" queue movement method can be used in the design of network devices, such as routers, chips with multiple FIFO queues and other software or hardware devices where element loss is permitted but unwanted. Using this model, with given probabilities of the queues, you can choose the best queue representation method, for example, between the classical serial cyclic method and "One after another" method.

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

References

1. Sedgewick R. *Algorithms in C++. Parts 1–4*. Addison-Wesley Professional, 1998. 752 p.
2. Bollapragada V., Murphy C., White R. *Inside Cisco IOS Software Architecture*. Cisco Press, 2000. 240 p.
3. Knuth D. *The Art of Computer Programming. Vol. 1*. Addison-Wesley Professional, 1997. 672 p.
4. Kalachev A. V. *Mnogoiadernye protsessory* [Multicore Architectures]. Moscow, BINOM Publ., 2014. 247 p. (In Russian).
5. Aksenova E. A., Sokolov A. V., Drac 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).
6. 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.
7. 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.
8. Sokolov A. V., Drac A. V. Modeling of Several Methods of Representation of n FIFO-Queues in the Single-Level Memory. *Jevristicheskie algoritmy i raspredelennye vychislenija*, 2014, vol. 1, no. 1, pp. 40–52 (In Russian).
9. Sokolov A. V. About Memory Distribution for Two Stacks. *Avtomatizatsiia eksperimenta i obrabotki dannykh*, 1980, pp. 65–71 (In Russian).
10. Yao A. C. An Analysis of a Memory Allocation Scheme for Implementing Stacks. *SIAM Journal on Computing*, 1981, vol. 10, pp. 398–403.
11. 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.
12. Louchard G., Schott R. Probabilistic Analysis of Some Distributed Algorithms. *Lecture Notes in Computer Science*, 1990, vol. 431, pp. 239–253.
13. 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.
14. Maier R. S. Colliding Stacks: A Large Deviations Analysis. *Random Structures and Algorithms*, 1991, no. 2, pp. 379–421.
15. Barkovsky E. A., Sokolov A. V. Optimal Control of Two Parallel FIFO Queues on an Infinite Time. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2015, no. 5, pp. 65–71 (In Russian). doi:10.15217/issn1684-8853.2015.5.65
16. Sokolov A. V. *Matematicheskie modeli i algoritmy optimal'nogo upravlenija dinamicheskimi strukturami dannyh* [Mathematical Models and Algorithms of Optimal Control of Dynamic Data Structures]. Petrozavodsk, PetrSU Publ., 2002. 216 p. (In Russian).
17. Sokolov A. V., Barkovsky E. A. Some Problems of Optimal Control of Two Parallel FIFO-queues. *Proc. 12th Int. Conf. "International Conference on Numerical Analysis and Applied Mathematics"*, AIP Publishing, 2015, vol. 1648, pp. 520003.
18. Kemeny J. G., Snell J. L. *Finite Markov Chains*. Van Nostrand, 1969. 210 p.
19. Tanenbaum A. S., Wetherall D. J. *Computer Networks*. Pearson Education, 2011. 933 p.