Министерство образования Российской Федерации
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра теоретических основ информатики

УДК 681.324

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, профессор, д.т.н.,
_____ Ю.Л. Костюк
«____» _____ 2003г

Биматов Дмитрий Владимирович

МОДЕЛИРОВАНИЕ ПОДСИСТЕМ ПАМЯТИ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Дипломная работа

Научный руководитель
профессор, д.т.н. _____ С.П. Сущенко,
Исполнитель,
студент V курса, гр. 1481 _____ Д.В. Биматов

Электронная версия дипломной работы помещена
в электронную библиотеку. Файл
Администратор

ТОМСК-2003

РЕФЕРАТ

Дипломная работа 50 с., 22 рис., 20 источника, 2 прил.

ПОДСИСТЕМА ПАМЯТИ, МНОГОПРОЦЕССОРНАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА, ИЕРАРХИЧЕСКАЯ ПАМЯТЬ, МНОГОУРОВНЕВАЯ ПАМЯТЬ, НЕБЛОКИРУЕМЫЙ КЭШ, ОПЕРАЦИОННЫЕ ХАРАКТЕРИСТИКИ, МАРКОВСКАЯ ЦЕПЬ, СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ, СИСТЕМА УРАВНЕНИЙ РАВНОВЕСИЯ

Объект исследования – двухуровневые подсистемы памяти симметричных многопроцессорных вычислительных систем

Цель исследования – исследовать влияние параметров двухуровневой подсистемы памяти на ее производительность

Метод исследования – аналитический, экспериментальный (на ЭВМ)

Результаты работы

- предложена математическая вероятностная модель функционирования двухуровневой подсистемы памяти для симметричной многопроцессорной системы;
- найдены общие подходы для вычисления операционных характеристик подсистемы памяти симметричной многопроцессорной системы;
- получены аналитические зависимости для вычисления вероятностей состояний некоторых подсистем памяти;
- исследовано влияние параметров подсистемы памяти на ее операционные характеристики.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ВС – вычислительная система;

SMP-система (Symmetrical MultiProcessing) – симметричная многопроцессорная система или система с разделяемой памятью;

ЦП – центральный процессор;

ОЗУ - оперативное запоминающее устройство (оперативная память);

КНТ – кэш неблокирующего типа;

КБТ – кэш блокирующего типа;

СМО – система массового обслуживания;

СУР – система уравнений равновесия.

ВВЕДЕНИЕ

В настоящее время в различных сферах человеческой деятельности активно используются вычислительная техника. Производительность вычислительных систем постоянно растет. Однако потребности в обработке и хранении все возрастающего количества информации так и остаются не полностью удовлетворенными. Эти потребности практически безграничны. Поэтому производители компьютерной техники постоянно ищут новые и перспективные способы увеличения производительности предлагаемых систем.

Существует два концептуальных подхода к увеличению производительности вычислительных систем.

Экстенсивный подход заключается в совершенствовании технологических процессов полупроводниковой промышленности, что позволяет увеличивать количество элементов, тактовые частоты микросхем. По так называемому «закону Мура» (Moore's Law), который действует уже более трех десятилетий, количество транзисторов на кристалле удваивается каждые 18 месяцев. Существенно ускорить этот процесс в настоящий момент и в ближайшем будущем не представляется возможным. Более того, в последнее время возникает опасность приближения к физическим барьерам уменьшения размера базового элемента микросхем.

Интенсивный подход заключается в совершенствовании архитектуры вычислительных систем, в обеспечении сбалансированной работы и эффективного взаимодействия всех узлов вычислительной установки для достижения наибольшей производительности. Работы в этом направлении приобретают все б**о**льшую значимость.

В 1946 году один из разработчиков первой электронной вычислительной машины «ENIAC» Джон фон Нейман (1903-1957гг) в своей работе «Предварительное обсуждение логической конструкции электронной вычислительной машины» предложил принципиальную архитектуру вычислительной системы. Концептуальная схема вычислительной системы приведена на рис.1. Любая вычислительная система состоит из следующих функционально независимых частей [1, 2]:

- Арифметико-логическое устройство;
- Устройство управления;
- Устройство памяти;
- Устройство ввода;

- Устройство вывода.

Арифметико-логическое устройство в комплексе с Устройством управления называют *процессором*. Процессор последовательно извлекает команды программы из памяти и выполняет определяемые ими операции. Вся информация, доступная компьютеру хранится в памяти. Взаимодействие компьютера с внешним миром происходит с помощью устройств ввода-вывода.

Со времен появления первой ЭВМ и до настоящего времени проблема оперативного снабжения центрального процессора командами и данными не теряет своей актуальности. Дело в том, что разрыв скорости обработки данных в центральном процессоре и скорости доступа к адресуемым операндам в оперативной памяти составляет несколько порядков [3, 4]. В связи с этим, очевидно, что высокое быстродействие вычислительной системы в целом в значительной мере определяется организацией ее подсистемы памяти [5, 6].

В последнее время широкое распространение получили многопроцессорные вычислительные системы. С 1980 года на рынке присутствуют симметричные мультипроцессорные системы (SMP-системы), в которых несколько процессоров разделяют одну физическую память, соединенную с ними с помощью разделяемой шины [7, 8]. Схема четырехпроцессорной SMP-системы приведена на рис.2. Достаточно высокая производительность и относительно низкая стоимость таких систем в значительной степени определили их популярность.

Однако общая системная шина и общая оперативная память становятся «узким горлом» SMP-системы, так как обращение одного процессора к общей ОЗУ заставляет всех остальных процессоров, желающих получить доступ к ОЗУ, ожидать первого. В этой ситуации задача построения эффективной архитектуры многоуровневой подсистемы памяти, минимизирующей обращения к основной памяти, становится наиболее актуальной.

Данная работа посвящена исследованию эффективности двухуровневых подсистем памяти симметричных многопроцессорных систем. Предполагается исследовать влияние основных параметров подсистемы памяти SMP-системы на ее производительность.

Это позволит решать задачу количественного определения производительности вычислительной системы по ее основным параметрам, а также оптимизации архитектуры ВС для обеспечения требуемой производительности.

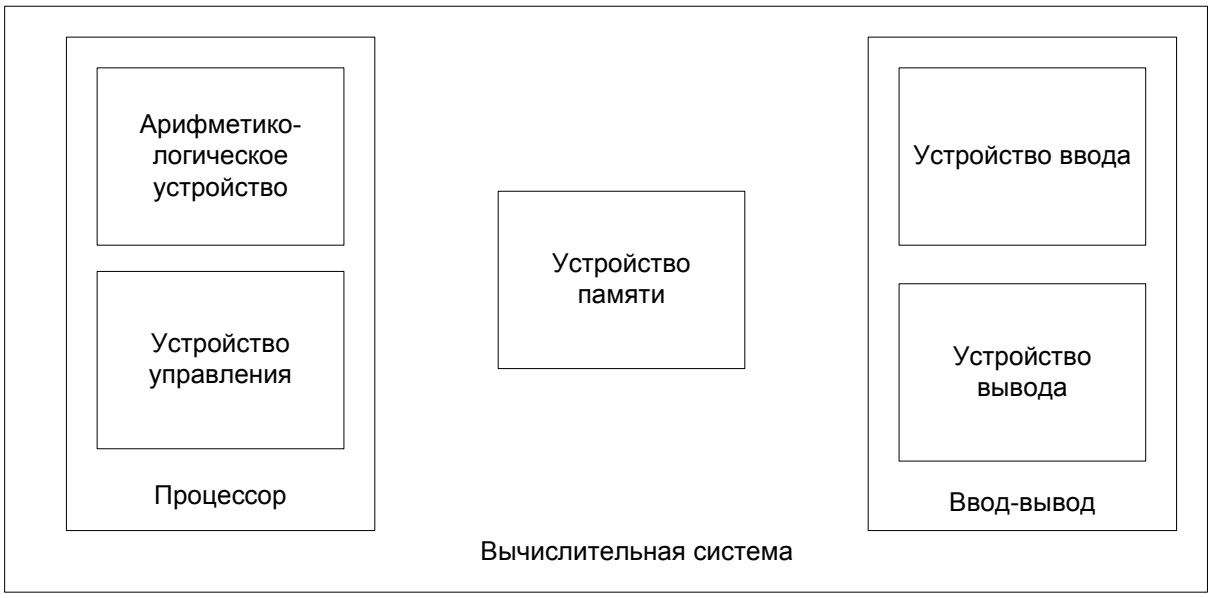

Рис. 1. Концептуальная схема вычислительной системы

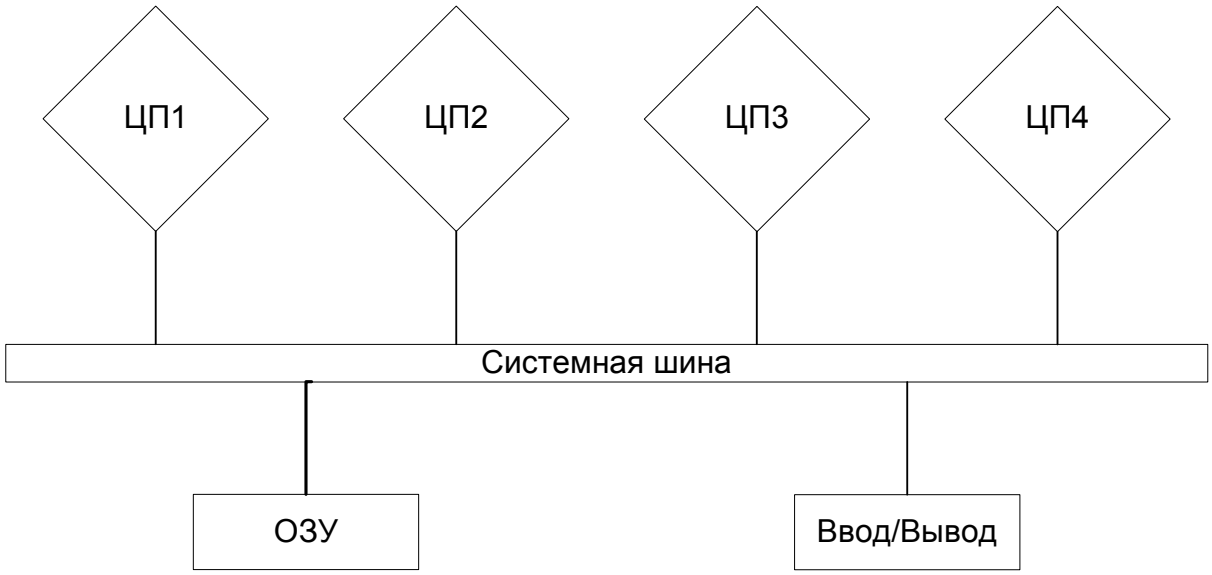

Рис. 2. Схема четырехпроцессорной SMP-системы

1. МНОГОУРОВНЕВАЯ ОРГАНИЗАЦИЯ ПАМЯТИ

В настоящее время при разработке вычислительных систем и компьютерных сетей для обеспечения эффективного доступа к данным широко используется многоуровневая организация памяти [9]. На верхнем уровне находится самая быстрая, дорогая и наименее емкая память, а на нижнем - самая медленная, дешевая и наиболее емкая. Память верхних уровней принято называть кэшем (рис. 3). Благодаря многоуровневой организации памяти возможно использовать механизм кэширования – накопление и сохранение часто используемых данных в верхних уровнях памяти для быстрого доступа к ним [10, 11, 12].

Многоуровневая организация памяти основывается на двух принципах [13]:

- *Принцип временной локальности*: данные, необходимые процессору в данный момент, могут потребоваться вновь через некоторое время;
- *Принцип пространственной локальности*: если процессор запрашивает данные по некоторому адресу, то, возможно, вскоре ему потребуются данные по этому же адресу с некоторым смещением.

Существуют два варианта реализации многоуровневого кэша:

- *Включающая архитектура*. Кэш более высокого уровня является источником данных для кэша менее высокого уровня. В этом случае кэш более высокого уровня гарантированно содержит все блоки кэша менее высокого уровня.
- *Исключающая архитектура*. Все уровни кэша используют ОЗУ в качестве источника кэширования и не содержат дубликатов.

В данной работе исследуется включающая архитектура процессорного кэша.

1.1. Функционирование многоуровневой памяти

Рассмотрим работу двухуровневой подсистемы памяти «Кэш-ОЗУ» (рис. 4). При поступлении запроса от ЦП на выборку адресуемого элемента памяти возможно возникновение одной из двух ситуаций:

а) «Попадание» (hit) – адресуемый элемент находится в кэше: кэш предоставляет процессору требуемый элемент;

b) «Промах» (miss) – адресуемый элемент в кэше не найден. В этом случае кэш обращается за требуемым элементом в ОЗУ, а затем предоставляет его процессору.

В промежуток времени между обращением к ОЗУ и возвратом элемента процессору кэш находится в заблокированном состоянии. Это означает, что кэш не может принимать других запросов от ЦП и процессору приходится ожидать. Так возникают нежелательные простои процессора при обращении к подсистеме памяти.

1.2. Понятие кэша неблокирующего типа

Для уменьшения нежелательных простоев процессора в подсистемах памяти современных вычислительных систем используется *кэш неблокирующего типа*.

Понятие неблокируемости в компьютерной терминологии раскрывается следующим образом. *Неблокируемый* – позволяющий выполнять последующие операции, даже если выполнение предшествующей операции (или предшествующих операций) не может быть полностью завершено [7].

При глубине неблокируемости N кэш имеет специальный буфер ёмкостью N элементов для хранения запросов к ОЗУ. Если количество транзакций, обрабатываемых в фазе обращения к ОЗУ, достигает N, то буфер запросов оказывается заполненным и кэш-память блокируется. Так, кэш с коэффициентом неблокируемости $N = 1$ является кэшем блокирующего типа (КБТ), а кэш с коэффициентом неблокируемости $N \geq 2$ является кэшем неблокирующего типа (КНТ).

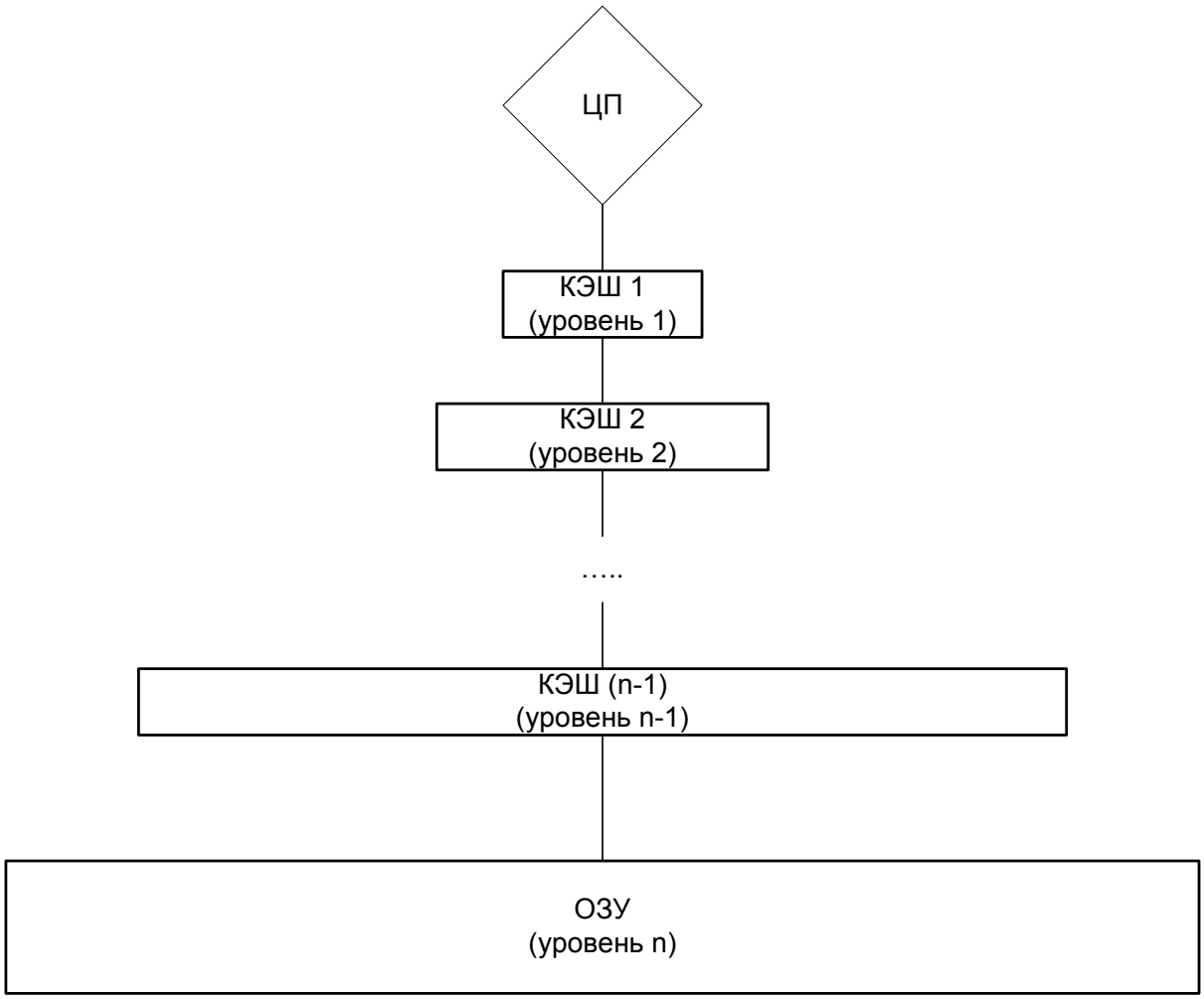

Рис. 3. Многоуровневая организация памяти

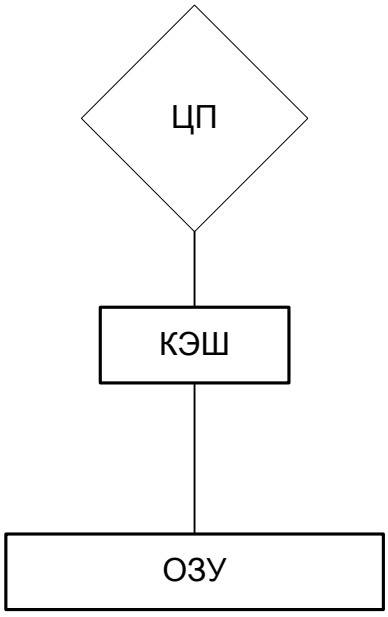

Рис. 4. Двухуровневая подсистема памяти «Кэш-ОЗУ»

2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

2.1. Описание модели

Рассмотрим многопроцессорную вычислительную систему с общей разделяемой оперативной памятью. Будем полагать, что процессорами порождается неограниченный поток обращений к подсистеме двухуровневой памяти. При этом процесс функционирования подсистемы памяти может быть описан работой двухстадийного конвейера [9].

На первой фазе работы конвейера выполняется обращение к кэш-памяти. Длительность этой фазы равна времени доступа к кэшу *t*. При попадании адресуемого объекта в кэш выполняется следующий запрос к иерархической памяти. В случае промаха одновременно происходит обработка текущего запроса на второй фазе – фазе доступа к оперативной памяти и следующей транзакции – на первой фазе. Таким образом, факт обработки транзакции доступа к иерархической памяти на второй фазе является случайным событием. Время обработки запроса на второй фазе в *K* раз превышает время обработки запроса на первой фазе и составляет $K \cdot t$ [14, 15].

При глубине неблокируемости N>0 подсистема многоуровневой памяти имеет буферы емкостью по *N* элементов для хранения запросов от каждого ЦП к оперативной памяти, которые последовательно обрабатываются элементами управления памятью на второй фазе конвейера. Если количество транзакций от процессора, обрабатываемых в фазе обращения к оперативной памяти, совпадает с коэффициентом неблокируемости *N*, то кэш-память этого ЦП оказывается заблокированной. В заблокированном состоянии первая стадия конвейера не работает. Разблокирование первой стадии наступает при завершении обработки одной транзакции на второй фазе конвейера.

Фаза доступа к оперативной памяти вычислителя может быть описана СМО с дискретным временем, многоэтапным обслуживанием и конечным накопителем. Функционирование данной СМО в станционарном состоянии задается цепью Маркова в *M*-мерном пространстве, где *M* – количество процессоров в ВС. Время между поступлениями заявок от *m*-того ЦП (транзакций обращения к ОЗУ) кратно *t* и имеет геометрическое распределение с параметром, равным вероятности промаха в кэш данного процессора R_m. Для завершения одной транзакции ОЗУ требуется выполнить *K* этапов [16, 17].

Итак, выделяются следующие параметры моделируемой подсистемы памяти:

- Число процессоров в вычислительной системе (*M)*;
- Глубина неблокируемости кэша каждого процессора (*N)*;
- Время выбора элемента из ОЗУ *K* (в тактах обращения к кэшу *t*);
- Вектор вероятностей промаха $\vec{R} = (R_1, R_2, ...R_M)$, где R_m - вероятность промаха в кэш *m*-того ЦП.

2.2. Состояния системы

Состояние системы обозначается вектором из *M* элементов $(i_1, i_2, ...i_m, ..., i_M)$, где i_m – количество этапов для обработки на второй фазе конвейера от *m*-того процессора.

Так как количество этапов для обработки на второй фазе конвейера от одного ЦП колеблется от 0 до $N \cdot K$, то потенциально цепь Маркова может иметь до $(NK+1)^M$ состояний. Однако не все эти состояния являются *корректными*. Это связано с тем, что по предположению модели, ОЗУ в каждый времени может выполнять транзакцию только от одного ЦП. Это означает, что корректное состояние имеет не более одной компоненты, не делящейся нацело на *K*.

Поэтому количество корректных состояний складывается из состояний, в которых все *M* компонент кратны *K*, и состояний, в которых (*M*-1) компонента кратна *K*, а одна компонента – не кратна *K*:

$$nStates = (N+1)^M + (N+1)^{M-1} \cdot M \cdot N \cdot (K-1). \qquad (2.1)$$

Некорректные состояния далее не рассматриваются.

Состояние $(0,...,0)$ будем называть *свободным*.

Состояния $(i_1, ...i_M)$, $0 \le i_m \le (N-1)K$ будем называть *m-незаблокированными*. В этих состояниях кэш *m*-того процессора является незаблокированным и ОЗУ может принимать запросы от *m*-того ЦП. Состояния $(i_1, ...i_M)$, $\forall m \ 0 \le i_m \le (N-1)K$ будем называть *полностью незаблокированными*. В этих состояниях ОЗУ может принимать запросы от любого ЦП.

В состояниях $(i_1, ...i_M)$, $(N-1)K+1 \le i_m \le NK$ кэш *m*-того процессора является заблокированным – назовем их *m-заблокированными*. Состояния $(i_1, ...i_M)$, $\forall m \ (N-1)K+1 \le i_m \le NK$ будем называть *полностью заблокированными*. В полностью заблокированных состояниях транзакции от ЦП не поступают, так как кэши всех процессоров заблокированы.

В свободном состоянии СМО простаивает, так как ОЗУ не имеет этапов для исполнения. Во всех остальных состояниях $(j_1,...j_M)$ параллельно с возможным поступлением запросов от процессоров обязательно происходит *откат*, то есть обработка одного этапа активной транзакции, $\forall m \neq a \;\; j'_m = j_m, \;\; j'_a = j_a - 1$, где a – номер активной транзакции. Номер активной транзакции определяется следующим образом. Если состояние $(j_1,...j_M)$ имеет компоненту j_a не кратную *K,* тогда a – номер активной транзакции. Если состояние $(j_1...j_M)$ имеет только кратные *K* компоненты, то номер активной транзакции равновероятно выбирается между $1 < l \le M$ ненулевыми компонентами состояния.

Вероятности переходов из состояния $(j_1,...j_M)$ в измененное состояние $(i_1,...i_M)$ вычисляются по следующей формуле:

$$p\left[(j_1,...j_M);\;\;(i_1,...i_M)\right] = \frac{1}{l}\cdot\prod_{m=1}^{M} A_m \text{, где } A_m = \begin{cases} 1, & j_M > (N-1)\cdot K \\ (1-R_m), & i_m = j'_m \\ R_m, & i_m = j'_m + K \end{cases}, \qquad (2.2)$$

где $1 < l \le M$ – количество возможных откатов из состояния $(j_1,...j_M)$.

Количество переходов из свободного состояния составляет 2^M. Максимальное количество переходов из одного состояния составляет

$$nJumps = M \cdot 2^M \qquad (2.3).$$

2.2. Вероятности состояний системы

Обозначим вероятности состояний системы как $P(i_1,...i_M)$. Тогда описанной цепи Маркова соответствует СУР вида:

$$\left\{ P(i_1,...i_M) = \sum_{\forall k,l} p\left[(j_1,...j_m);(i_1,...i_m)\right]\cdot P(j_1,...j_M), \right. \qquad (2.4)$$

где $p\left[(j_1,...j_m);(i_1,...i_m)\right]$ - вероятность перехода из состояния $(j_1,...j_M)$ в состояние $(i_1,...i_M)$ [18, 19].

Таким образом, исследование Марковской СМО, моделирующей поведение подсистемы памяти, сводится к решению системы линейных уравнений вида (2.4). Заметим, что одно из уравнений системы (2.4) обязательно будет линейно зависеть от остальных. Поэтому для полного решения системы требуется использовать условие нормировки: $\sum P(i_1,...i_M) = 1$.

3. ВЫЧИСЛЕНИЕ ВЕРОЯТНОСТЕЙ СОСТОЯНИЙ

Для наиболее полного описания функционирования Марковской СМО требуется вычислить вероятности ее состояний. Данная глава посвящена аналитическому расчету вероятностей состояний для некоторых двухуровневых подсистем памяти. Отметим, что аналитически найденные вероятности состояний системы позволяют аналитически вычислить основные операционные характеристики системы.

Также рассматриваются способы численного вычисления вероятностей состояний для двухуровневой подсистемы памяти с произвольными значениями параметров M, N, K, $\vec{R}$.

3.1. Однопроцессорные вычислительные системы

Рассмотрим однопроцессорную вычислительную систему (M=1), основанную на кэше блокирующего типа (N=1). Граф переходов соответствующей цепи Маркова представлен на рис. 5. Система уравнений равновесия данной системы выглядит следующим образом:

$$\begin{aligned} &P_0 = (1-R)\cdot P_0 + P_1; \\ &P_1 = P_2; \\ &\ldots \\ &P_{K-1} = P_K; \\ &P_K = R\cdot P_0. \end{aligned} \qquad (3.1)$$

Решение СУР (3.1) представляется в виде:

$$P_0 = \frac{1}{1+KR}; \qquad P_i = \frac{R}{1+KR},\ i = \overline{1,K}.$$

Теперь рассмотрим подсистему памяти однопроцессорной вычислительной системы с параметрами N=2, K=2. Граф переходов соответствующей цепи Маркова представлен на рис. 6. Система уравнения равновесия для данной системы выглядит следующим образом:

$$\begin{aligned} &P_0 = (1-R)\cdot P_0 + (1-R)\cdot P_1; \\ &P_1 = (1-R)\cdot P_2; \\ &P_2 = R\cdot P_0 + R\cdot P_1 + P_3; \end{aligned} \qquad (3.2)$$

$$P_3 = R \cdot P_2 .$$

Решение СУР (3.2) представляется в виде:

$$P_0 = \frac{(1-R)^2}{1+R^2};$$

$$P_1 = \frac{R-R^2}{1+R^2};$$

$$P_2 = \frac{R}{1+R^2};$$

$$P_3 = \frac{R^2}{1+R^2}.$$

3.2. Двухпроцессорные вычислительные системы

Рассмотрим двухпроцессорную вычислительную систему (M=2), основанную на кэше блокирующего типа (N=1). Граф переходов соответствующей цепи Маркова представлен на рис. 7.

Переходные вероятности Π_{ij}^{lk} из одного состояния (i, j) в измененное состояние (l, k) имеют вид:

$$\Pi_{ij}^{lk} = \begin{cases} R_1(1-R_2), i=0,\ j=0, l=K, k=0; \\ 1-R_2, i=\overline{1,K},\ j=0, l=i-1, k=0; \\ R_2, i=\overline{1,K},\ j=0, l=i-1, k=K; \\ R_2(1-R_1), i=0,\ j=0, l=0, k=K; \\ 1-R_1, i=0,\ j=\overline{1,K}, l=0, k=j-1; \\ R_1, i=0,\ j=\overline{1,K}, l=K, k=j-1; \\ R_1R_2, i=0,\ j=0, l=K, k=K; \\ q, i=K,\ j=K, l=K-1, k=K; \\ 1-q, i=K,\ j=K, l=K, k=K-1; \\ 1, i=\overline{1,K-1},\ j=K, l=i-1, k=K; \\ 1, i=K,\ j=\overline{1,K-1}, l=K, k=j-1. \end{cases}$$

Система уравнений равновесия для данной системы выглядит следующим образом:

$$P_{ij} = \sum_{\forall k,l} \Pi_{kl}^{ij} \cdot P_{kl} . \qquad (3.3)$$

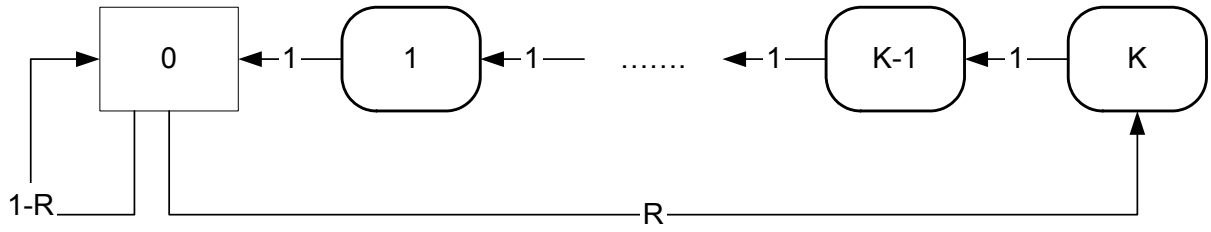

Рис. 5. Марковская цепь для подсистемы памяти однопроцессорной ВС (N=1)

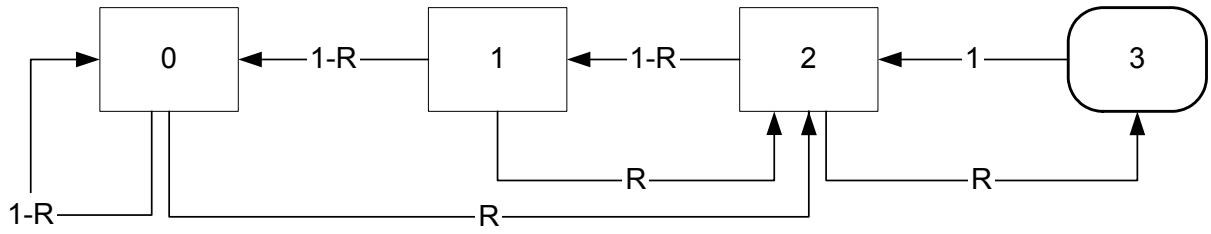

Рис. 6. Марковская цепь для подсистемы памяти однопроцессорной ВС (N=2; K=2)

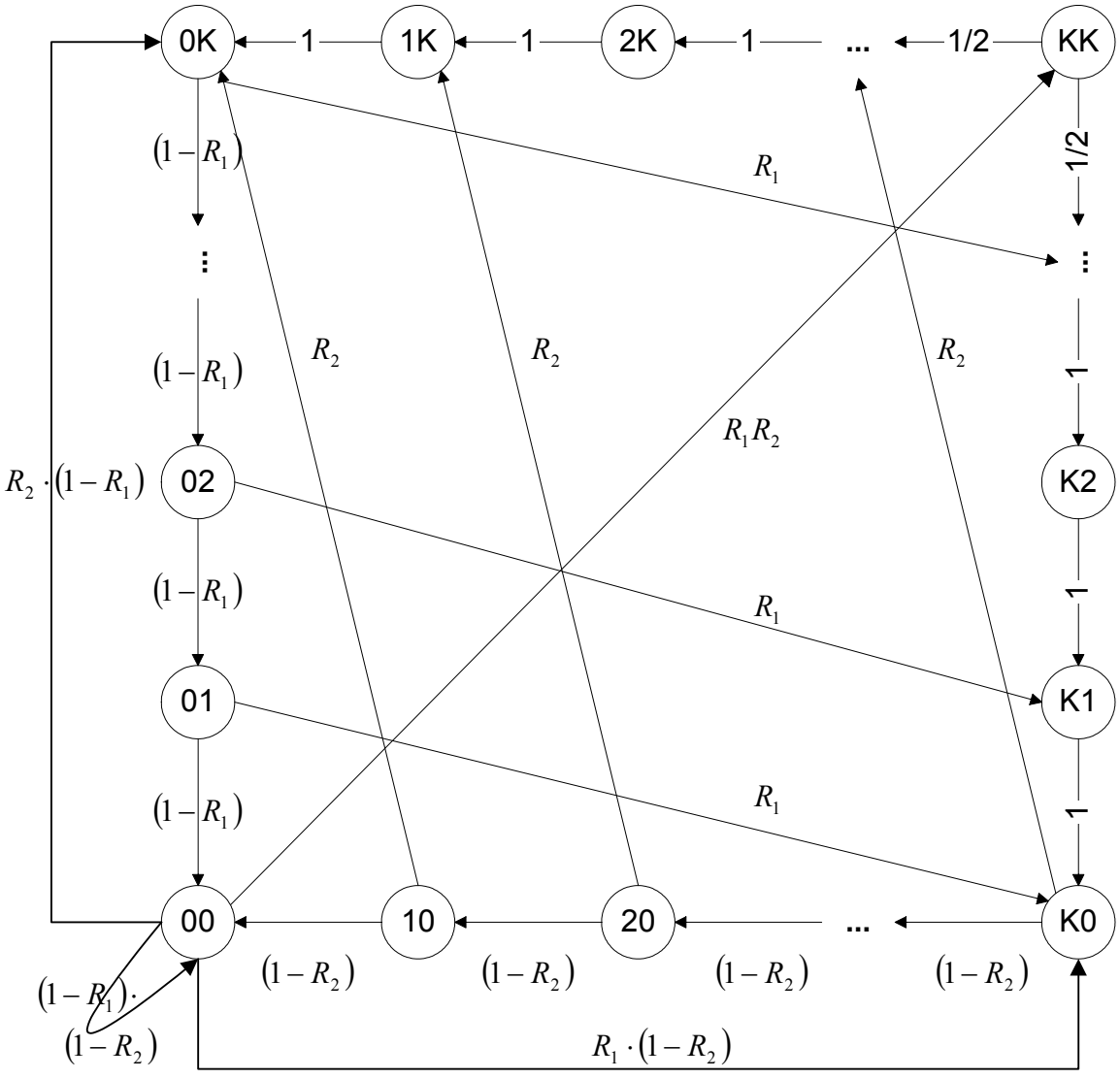

Рис. 7. Марковская цепь для двухпроцессорной подсистемы памяти (N=1; K=2)

Решение СУР (3.3) представляется в виде [16]:

$$P_{i0} = P_{K0} \cdot (1-R_2)^{K-i},\ i = \overline{1, K-1};$$

$$P_{K0} = \frac{1}{z} \cdot \left[R_1 + R_2 - R_1 R_2 - R_2 (1-R_1)^K \left(1 - R_1 (1-q)\right)\right];$$

$$P_{jK} = P_{KK} \cdot q + P_{K0} \cdot \left(1 - (1-R_2)^{K-j}\right),\ j = \overline{1, K-1};$$

$$P_{0j} = P_{0K} \cdot (1-R_1)^{K-j},\ j = \overline{1, K-1};$$

$$P_{0K} = \frac{1}{z} \cdot \left[R_1 + R_2 - R_1 R_2 - R_1 (1-R_2)^K (1 - R_2 q)\right]; \quad (3)$$

$$P_{Kj} = P_{KK} \cdot (1-q) + P_{0K} \cdot \left(1 - (1-R_1)^{K-j}\right),\ j = \overline{1, K-1};$$

$$P_{KK} = P_{00} \cdot R_1 R_2;$$

$$P_{00} = \frac{1}{z} \cdot \left[(1-R_1)^K + (1-R_2)^K - (1-R_1)^K (1-R_2)^K\right],$$

где

$$z = 2K \cdot \left[R_1 + R_2 (1-R_1)\right] + (1-R_1)^K \cdot \left[1 - KR_2 + KR_1 R_2 (2-q)\right] +$$
$$+ (1-R_2)^K \cdot \left[1 - KR_1 + KR_1 R_2 (1+q)\right] - (1-R_1)^K \cdot (1-R_2)^K \left[1 + KR_1 R_2\right].$$

Теперь рассмотрим подсистему памяти с параметрами M=2, N=2, K=2. Граф переходов соответствующей цепи Маркова представлен на рис. 8. Система уравнений равновесия для данной системы выглядит следующим образом:

$$P_{00} = (1-R_1)(1-R_2) \cdot \left[P_{00} + P_{10} + P_{01}\right];$$

$$P_{10} = (1-R_1)(1-R_2) \cdot P_{20};$$

$$P_{01} = (1-R_1)(1-R_2) \cdot P_{02};$$

$$P_{20} = (1-R_2) \cdot P_{30} + R_1 \cdot (1-R_2) \cdot \left[P_{00} + P_{10} + P_{01}\right] + (1-R_1)(1-R_2) \cdot P_{21};$$

$$P_{02} = (1-R_1) \cdot P_{03} + R_2 \cdot (1-R_1) \cdot \left[P_{00} + P_{10} + P_{01}\right] + (1-R_1)(1-R_2) \cdot P_{12};$$

$$P_{21} = \frac{(1-R_1)(1-R_2)}{2} \cdot P_{22} + R_1 (1-R_2) \cdot P_{02};$$

$$P_{12} = \frac{(1-R_1) \cdot (1-R_2)}{2} \cdot P_{22} + R_2 (1-R_1) \cdot P_{20};$$

$$P_{22} = R_2 P_{30} + R_1 P_{03} + (1-R_2) \cdot P_{32} + (1-R_1) \cdot P_{23} +$$
$$+ R_1 R_2 \cdot \left[P_{00} + P_{10} + P_{01}\right] +$$
$$+ R_1 (1-R_2) \cdot P_{12} + R_2 (1-R_1) \cdot P_{21}$$

$$P_{30} = (1-R_2) \cdot P_{40} + R_1 (1-R_2) \cdot P_{20}; \quad (3.4)$$

$$P_{03} = (1-R_1) \cdot P_{04} + R_2 (1-R_1) \cdot P_{02};$$

$$P_{40} = (1-R_2)\cdot P_{41} + R_1(1-R_2)\cdot P_{21};$$

$$P_{04} = (1-R_1)\cdot P_{14} + R_2(1-R_1)\cdot P_{12};$$

$$P_{41} = \frac{(1-R_2)}{2}\cdot P_{42} + \frac{R_1(1-R_2)}{2}\cdot P_{22};$$

$$P_{14} = \frac{(1-R_1)}{2}\cdot P_{24} + \frac{R_2(1-R_1)}{2}\cdot P_{22};$$

$$P_{32} = \frac{(1-R_2)}{2}\cdot P_{42} + R_1R_2P_{20} + \frac{R_1(1-R_2)}{2}\cdot P_{22} + R_2P_{40};$$

$$P_{23} = \frac{(1-R_1)}{2}\cdot P_{24} + R_1R_2P_{02} + \frac{R_2(1-R_1)}{2}\cdot P_{22} + R_1P_{04};$$

$$P_{42} = P_{43} + R_1R_2P_{21} + R_1P_{23} + R_2P_{41};$$

$$P_{24} = P_{34} + R_1R_2P_{12} + R_2P_{32} + R_1P_{14};$$

$$P_{43} = P_{34} = \frac{R_1R_2P_{22} + R_2P_{42} + R_1P_{24}}{2};$$

В однородном случае $(R_1 = R_2 = R)$ решение системы уравнений равновесия (3.4) представимо в виде:

$$P_{00} = \frac{(1-R)^6}{R^2}W;$$

$$P_{10} = P_{01} = \frac{(2-R)}{2R}W;$$

$$P_{20} = P_{02} = \frac{(1-R)^2(2-R)}{2R}W;$$

$$P_{21} = P_{12} = \frac{(1-R)^2}{2}Z + \frac{(1-R)^3(2-R)}{2}W;$$

$$P_{22} = Z;$$

$$P_{30} = P_{03} = \frac{R(1-R)^3(2-R)}{2}Z + \frac{(2-R)\left[R(1-R)^5 + (1-R)^3\right]}{2}W + \frac{(1-R)^3}{2}U;$$

$$P_{40} = P_{04} = \frac{R(1-R)^2(2-R)}{2}Z + \frac{R(1-R)^4(2-R)}{2}W + \frac{(1-R)^2}{2}U;$$

$$P_{32} = P_{23} = \frac{(1-R)\cdot\left(R^4 - 3R^3 + 2R^2 + R\right)}{2}Z +$$

$$+\frac{(1-R)^2\cdot\left(-R^5 + 4R^4 - 5R^3 + R^2 + 2R\right)}{2}W +$$

$$+\frac{(1-R)\cdot(-R^2 + R + 1)}{2}U;$$

$$P_{41} = P_{14} = \frac{R(1-R)}{2} Z + \frac{(1-R)}{2} U;$$

$$P_{42} = P_{24} = U;$$

$$P_{43} = P_{34} = \frac{R^2}{2} Z + R \cdot U,$$

где

$$D = \begin{pmatrix} 2R^{14} - 10R^{13} + 11R^{12} + 12R^{11} + 8R^{10} - \\ -120R^9 + 112R^8 + 148R^7 - 330R^6 + \\ +246R^5 - 111R^4 + 12R^3 + 36R^2 - 16R + 4 \end{pmatrix};$$

$$Z = R^2 (1-R)^2 (2-R^2) \cdot \frac{(2R^6 - 12R^5 + 24R^4 - 14R^3 - 9R^2 + 8R + 2)}{D};$$

$$W = R^2 (1-R)^2 (2-R^2) \cdot \frac{(-2R^4 + 6R^3 - 5R^2 + 2)}{D};$$

$$U = \frac{\left(-R^6 + 4R^5 - 4R^4 - 2R^3 + 4R^2\right)}{2} \cdot \left[Z + (1-R)^2 W\right].$$

В случае $R_2 = 1$ система уравнений равновесия данной системы преобразуется к виду:

$$\begin{aligned}
&P_{02} = (1-R_1) \cdot P_{03}; \\
&P_{03} = (1-R_1) \cdot P_{04} + (1-R_1) \cdot P_{02}; \\
&P_{04} = (1-R_1) \cdot P_{14}; \\
&P_{14} = \left(\frac{1-R_1}{2}\right) \cdot P_{24} + \left(\frac{1-R_1}{2}\right) \cdot P_{22}; \\
&P_{22} = (1-R_1) \cdot P_{23} + R_1 \cdot P_{03}; \\
&P_{23} = \left(\frac{1-R_1}{2}\right) \cdot P_{24} + \left(\frac{1-R_1}{2}\right) \cdot P_{22} + R_1 \cdot P_{02} + R_1 \cdot P_{04}; \\
&P_{24} = P_{34} + R_1 \cdot P_{14}; \\
&P_{42} = P_{43} + R_1 \cdot P_{23}; \\
&P_{43} = P_{34} = \frac{P_{42} + R_1 \cdot P_{22} + R_1 \cdot P_{24}}{2}.
\end{aligned} \qquad (3.5)$$

Решение системы (3.5) выглядит так:

$$P_{02} = \frac{(1-R_1)^4}{Z};$$

$$P_{03} = \frac{(1-R_1)^3}{Z};$$

$$P_{04} = \frac{R_1(1-R_1)^2 \cdot (2-R_1)}{Z};$$

$$P_{14} = \frac{R_1(1-R_1) \cdot (2-R_1)}{Z};$$

$$P_{22} = \frac{R_1(1-R_1)^2 (4-3R_1)}{Z};$$

$$P_{23} = \frac{R_1(1-R_1) \cdot (3-2R_1)}{Z};$$

$$P_{24} = \frac{R_1^2 \cdot (3R_1^2 - 10R_1 + 9)}{Z};$$

$$P_{42} = \frac{2R_1^2 \cdot (2R_1^2 - 6R_1 + 5)}{Z};$$

$$P_{43} = P_{34} = \frac{R_1^2 \cdot (2R_1^2 - 7R_1 + 7)}{Z},$$

где

$$Z = 8R_1^4 - 24R_1^3 + 18R_1^2 + 4R_1 + 2.$$

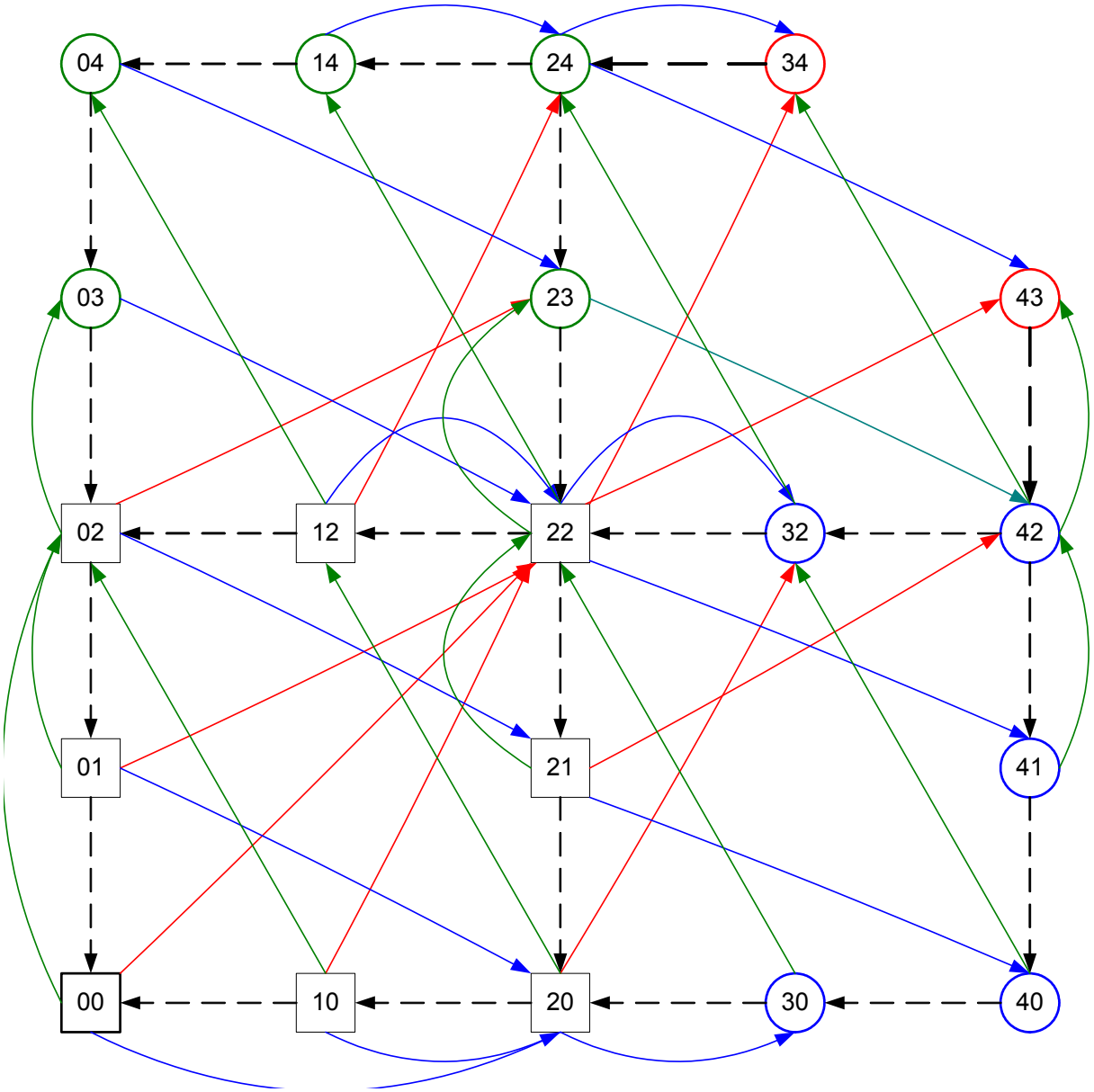

Рис. 8. Марковская цепь для двухпроцессорной подсистемы памяти на основе кэша неблокирующего типа (N=2; K=2)

3.3. Трехпроцессорная вычислительная система

Рассмотрим подсистему памяти трехпроцессорной вычислительной системы (M=3) с кэшем блокирующего типа (N=1). При K=2 СУР данной системы выглядит следующим образом:

$$
\begin{aligned}
P_{000} &= (1-R_1)(1-R_2)(1-R_3)P_{000} + \\
&\quad + (1-R_2)(1-R_3)P_{100} + (1-R_1)(1-R_3)P_{010} + (1-R_1)(1-R_2)P_{001};\\
P_{100} &= (1-R_2)(1-R_3)P_{200};\\
P_{010} &= (1-R_1)(1-R_3)P_{020};\\
P_{001} &= (1-R_1)(1-R_2)P_{002};\\
P_{200} &= R_1(1-R_2)(1-R_3)P_{000} + R_1(1-R_3)P_{010} + R_1(1-R_2)P_{001} + (1-R_3)P_{210} + (1-R_2)P_{201};\\
P_{020} &= R_2(1-R_1)(1-R_3)P_{000} + R_2(1-R_3)P_{100} + R_2(1-R_1)P_{001} + (1-R_3)P_{120} + (1-R_1)P_{021};\\
P_{002} &= R_3(1-R_1)(1-R_2)P_{000} + R_3(1-R_2)P_{100} + R_3(1-R_1)P_{010} + (1-R_2)P_{102} + (1-R_1)P_{012};\\
P_{210} &= \frac{1}{2}(1-R_3)P_{220} + R_1(1-R_3)P_{020};\\
P_{120} &= \frac{1}{2}(1-R_3)P_{220} + R_2(1-R_3)P_{200};\\
P_{201} &= \frac{1}{2}(1-R_2)P_{202} + R_1(1-R_2)P_{002};\\
P_{102} &= \frac{1}{2}(1-R_2)P_{202} + R_3(1-R_2)P_{200};\\
P_{021} &= \frac{1}{2}(1-R_1)P_{022} + R_2(1-R_1)P_{002};\\
P_{012} &= \frac{1}{2}(1-R_1)P_{022} + R_3(1-R_1)P_{020};\\
P_{220} &= R_1R_2(1-R_3)P_{000} + R_1R_2P_{001} + R_2P_{201} + R_1P_{021} + P_{221};\\
P_{202} &= R_1R_3(1-R_2)P_{000} + R_1R_3P_{010} + R_3P_{210} + R_1P_{012} + P_{212};\\
P_{022} &= R_2R_3(1-R_1)P_{000} + R_2R_3P_{100} + R_2P_{102} + R_3P_{120} + P_{122};\\
P_{221} &= \frac{1}{3}P_{222} + R_1R_2P_{002} + \frac{R_1}{2}P_{022} + \frac{R_2}{2}P_{202};\\
P_{212} &= \frac{1}{3}P_{222} + R_1R_3P_{020} + \frac{R_1}{2}P_{022} + \frac{R_3}{2}P_{220};
\end{aligned}
\qquad (3.6)
$$

$$P_{122} = \frac{1}{3} P_{222} + R_2 R_3 P_{200} + \frac{R_2}{2} P_{202} + \frac{R_3}{2} P_{220};$$

$$P_{222} = R_1 R_2 R_3 P_{000}.$$

В однородном случае $(R_1 = R_2 = R_3 = R)$ система уравнений равновесия (3.6) преобразуется к виду:

$$\begin{aligned}
&\left(1-(1-R)^3\right)\cdot P_{000} = (1-R)^2 \cdot \left[P_{100} + P_{010} + P_{001}\right];\\
&\left[P_{100} + P_{010} + P_{001}\right] = (1-R)^2 \cdot \left[P_{200} + P_{020} + P_{002}\right];\\
&\left[P_{200} + P_{020} + P_{002}\right] = 3R(1-R)^2 \cdot P_{000} + 2R(1-R)\cdot\left[P_{100} + P_{010} + P_{001}\right] +\\
&\qquad + (1-R)\cdot\left[P_{210} + P_{201} + P_{120} + P_{021} + P_{102} + P_{012}\right];\\
&\left[P_{210} + P_{201} + P_{120} + P_{021} + P_{102} + P_{012}\right] = \qquad (3.7)\\
&\qquad = (1-R)\cdot\left[P_{220} + P_{202} + P_{022}\right] + 2R(1-R)\cdot\left[P_{200} + P_{020} + P_{002}\right];\\
&\left[P_{220} + P_{202} + P_{022}\right] = 3R^2(1-R)\cdot P_{000} + R^2 \cdot\left[P_{100} + P_{010} + P_{001}\right] +\\
&\qquad + R\cdot\left[P_{210} + P_{201} + P_{120} + P_{021} + P_{102} + P_{012}\right] + 1\cdot\left[P_{221} + P_{212} + P_{122}\right];\\
&\left[P_{221} + P_{212} + P_{122}\right] = P_{222} + R^2 \cdot\left[P_{200} + P_{020} + P_{002}\right] + R\cdot\left[P_{220} + P_{202} + P_{022}\right];\\
&P_{222} = R^3 \cdot P_{000};
\end{aligned}$$

Решение системы уравнений равновесия (3.7) будет выглядеть так:

$$P_{000} = \frac{(1-R)^6}{Z};$$

$$P_{100} = P_{010} = P_{001} = \frac{(1-R)^4 - (1-R)^7}{3Z};$$

$$P_{200} = P_{020} = P_{002} = \frac{(1-R)^2 - (1-R)^5}{3Z};$$

$$P_{210} = P_{201} = P_{120} = P_{021} = P_{102} = P_{012} = \frac{(1-R) - (2R+1)(1-R)^4 - R(1-R)^7}{6Z};$$

$$P_{220} = P_{202} = P_{022} = \frac{1 - 2R(1-R)^2 - (2R+1)(1-R)^3 + 2R(1-R)^5 - R(1-R)^6}{3Z};$$

$$P_{221} + P_{212} + P_{122} = \frac{R - R^2(1-R)^2 - (2R+1)R(1-R)^3 + R^2(1-R)^5 - R^2(1-R)^7}{3Z};$$

$$P_{222} = \frac{R^3(1-R)^6}{Z},$$

где

$$Z = 2 - \left(R^2 + 2R - 1\right)(1-R)^2 - (2R+1)(R+1)(1-R)^3 - 2R(1-R)^4 +$$

$$+\left(R^2+2R-1\right)(1-R)^5+R^3(1-R)^6-\left(R^2+R\right)(1-R)^7;$$

В случае $(R_2=R_3=1)$ система уравнений равновесия (3.6) преобразуется в виду:

$$\begin{aligned}
&[P_{020}+P_{002}]=(1-R_1)\cdot[P_{021}+P_{012}];\\
&[P_{021}+P_{012}]=(1-R_1)\cdot[P_{020}+P_{002}]+(1-R_1)\cdot P_{022};\\
&[P_{220}+P_{202}]=R_1\cdot[P_{021}+P_{012}]+[P_{221}+P_{212}];\\
&P_{022}=P_{122}\\
&[P_{221}+P_{212}]=R_1\cdot[P_{020}+P_{002}]+\frac{1}{2}\cdot[P_{220}+P_{202}]+R_1\cdot P_{022};\\
&P_{122}=\frac{1}{2}\cdot[P_{220}+P_{202}];
\end{aligned} \qquad (3.8)$$

Решение СУР (3.8) будет выглядеть так:

$$\begin{aligned}
&P_{020}=P_{002}=\frac{(1-R_1)^2}{2Z};\\
&P_{021}=P_{012}=\frac{(1-R_1)}{2Z};\\
&P_{220}=P_{202}=\frac{R_1(2-R_1)}{Z};\\
&P_{022}=\frac{R_1(2-R_1)}{Z}\\
&P_{221}=P_{212}=\frac{R_1(3-R_1)}{2Z};\\
&P_{122}=\frac{R_1(2-R_1)}{Z},
\end{aligned}$$

где

$$Z=-4R_1^2+8R_1+2.$$

3.4. Четырехпроцессорная вычислительная система

Теперь рассмотрим подсистему памяти четырехпроцессорной вычислительной системы с кэшем блокирующего типа (N=1). В этом случае, а также в случаях с большим количеством ЦП, граф цепи Маркова в графическом виде трудно представим (M-мерное пространство).

В однородном случае $R_2 = R_3 = R_4 = 1$ система уравнений равновесия данной системы будет выглядеть следующим образом:

$$\begin{aligned}
&\left[P_{0220} + P_{0202} + P_{0022}\right] = \left(1 - R_1\right) \cdot \left[P_{0221} + P_{0212} + P_{0122}\right];\\
&\left[P_{0221} + P_{0212} + P_{0122}\right] = \left(1 - R_1\right) \cdot \left[P_{0220} + P_{0202} + P_{0022}\right] + \left(1 - R_1\right) \cdot P_{0222};\\
&\left[P_{2220} + P_{2202} + P_{2022}\right] = R_1 \cdot \left[P_{0221} + P_{0212} + P_{0122}\right] + \left[P_{2221} + P_{2212} + P_{2122}\right]; \qquad (3.9)\\
&P_{0222} = P_{1222};\\
&\left[P_{2221} + P_{2212} + P_{2122}\right] = R_1 \cdot \left[P_{0220} + P_{0202} + P_{0022}\right] + \frac{2}{3} \cdot \left[P_{2220} + P_{2202} + P_{2022}\right] + R_1 \cdot P_{0222};\\
&P_{1222} = \frac{1}{3} \cdot \left[P_{2220} + P_{2202} + P_{2022}\right].
\end{aligned}$$

Решение СУР (3.9) получается таким:

$$P_{0220} = P_{0202} = P_{0022} = \frac{\left(1 - R_1\right)^2}{3Z};$$

$$P_{0221} = P_{0212} = P_{0122} = \frac{\left(1 - R_1\right)}{3Z};$$

$$P_{2220} = P_{2202} = P_{2022} = \frac{1 - \left(1 - R_1\right)^2}{Z};$$

$$P_{0222} = \frac{1 - \left(1 - R_1\right)^2}{Z};$$

$$P_{2221} = P_{2212} = P_{2122} = \frac{5R_1 - 2R_1^2}{3Z};$$

$$P_{1222} = \frac{1 - \left(1 - R_1\right)^2}{Z},$$

где

$$Z = 2 + 12R_1 - 6R_1^2.$$

3.5. Численный расчет вероятностей состояний

Для вычисления вероятностей состояний систем с более высоким количеством процессоров *M* или коэффициентом неблокируемости *N*, или временем обращения к оперативной памяти *K* нахождение аналитического решения представляется достаточно трудоёмким. К тому же интерпретация громоздкого аналитического решения затруднена.

Существуют два способа численного вычисления вероятностей состояний любых Марковских цепей:

1. Численное решение. Заключается в решении СУР с учетом условия нормировки одним из известных численных методов решения СЛУ, например, методом Гаусса [20].
2. Итерационное решение. Заключается в последовательном умножении вектора вероятностей состояний на матрицу переходов. Таким образом, после n таких умножений мы найдем вектор вероятностей состояний на *n*-том такте системы:

$$\pi^{(n)} = P \cdot \pi^{(n-1)} \qquad \text{или} \qquad \pi^{(n)} = P^n \cdot \pi^{(0)}, \tag{3.10}$$

где $\pi^{(0)}$ - начальный вектор вероятностей состояний, $\pi^{(i)}$ - вектор вероятностей состояний системы после *i*-того такта, *P* – матрица переходов цепи Маркова.

Существует теорема, которая говорит о том, что для однородной, неприводимой и апериодической цепи Маркова всегда существует вектор предельных вероятностей состояний, не зависящий от начального вектора вероятностей состояний:

$$\pi = \lim_{n \to \infty} \left(P^n \cdot \pi^{(0)} \right), \ \forall \pi^{(0)}$$

То есть итеративный процесс (3.10) обязательно сойдется независимо от выбора начального вектора состояний [19]. Именно этот подход используется в разработанном автором приложении «SMPMemMark» для численного вычисления вероятностей состояний системы.

4. ВЫЧИСЛЕНИЕ ОПЕРАЦИОННЫХ ХАРАКТЕРИСТИК

Важнейшими операционными характеристиками многоуровневой подсистемы памяти являются:

- вероятность блокировки;
- среднее время выполнения запроса (транзакции доступа к памяти);
- пропускная способность.

4.1. Вероятность блокировки

Операционная характеристика «Вероятность блокировки» показывает, как часто кэш *m*-того процессора оказывается заблокированным. Вероятность блокировки кэш-памяти *m*-того процессора определяется как сумма вероятностей состояний системы, в которых кэш *m*-того процессора является заблокированным. Напомним, что блокировка КНТ происходит, когда количество обрабатываемых транзакций на втором этапе конвейера достигает глубины неблокируемости *N*, а выбор адресуемого элемента из ОЗУ занимает *K* этапов.

Тогда вероятность блокировки кэш-памяти *m*-того процессора может быть вычислена из вероятностей состояний системы следующим образом:

$$Q_m = \sum_{i_1=0}^{NK} \dots \sum_{i_m=(N-1)\cdot K+1}^{NK} \dots \sum_{i_M=0}^{NK} P(i_1,\dots i_M).$$

Вероятность блокировки кэш-памяти всех процессоров может быть вычислена следующим образом:

$$\overline{Q} = \sum_{i_1=(N-1)\cdot K+1}^{NK} \dots \sum_{i_M=(N-1)\cdot K+1}^{NK} P(i_1,\dots i_M).$$

4.2. Среднее время выполнения запроса

Операционная характеристика «Среднее время выполнения запроса» (или «Средняя задержка») показывает, сколько времени в среднем выполняется транзакция доступа к подсистеме памяти. При прочих равных условиях, чем меньшее время требуется для обработки отдельной транзакции, тем эффективнее функционирует

подсистема памяти. При достаточно большой средней задержке возникают ощутимые простои процессора.

Напомним, что транзакция от *m*-того процессора может быть успешно завершена уже на первом этапе конвейера при попадании к кэш (вероятность $1-R_m$). В таком случае, время выполнения транзакции будет равно времени обращения в кэш t. В случае промаха в кэш (вероятность R_m), время выполнения транзакции от *m*-того процессора складывается из времени разблокировки кэша $Tblock_m$ и времени обращения к ОЗУ $Taccess_m$:

$$T_m = (1-R_m) + R_m \cdot (Tblock_m + Taccess_m)$$

Используя формулу Литтла [19], получаем, что

$$Tblock_m = \frac{\overline{N}_{bm}}{\lambda_m},\ Taccess_m = \frac{\overline{N}_m}{\lambda_m},$$

где $\overline{N}_m$ - среднее количество этапов обработки транзакций обращения *m*-того процессора к иерархической памяти в фазе доступа к оперативной памяти, $\overline{N}_{bm}$ - среднее число этапов обслуживания, блокирующих доступ к кэш-памяти *m*-того процессора, а λ_m - интенсивность потока, принятого к обслуживанию от *m*-того процессора.

В общем случае величины $\overline{N}_m$ и $\overline{N}_{bm}$ соответственно определяются следующими соотношениями:

$$\overline{N}_m = \sum_{i_1=0}^{NK} \ldots \sum_{i_m=1}^{NK} \ldots \sum_{i_M=0}^{NK} \left[i_m \cdot P(i_1, \ldots i_M) \right];$$

$$\overline{N}_{bm} = \sum_{i_1=0}^{NK} \ldots \sum_{i_m=(N-1)\cdot K+1}^{NK} \ldots \sum_{i_M=0}^{NK} \left[\left(i_m - (N-1)\cdot K \right) \cdot P(i_1, \ldots i_M) \right].$$

Заметим, что для кэша блокирующего типа $(N=1)$ величины $\overline{N}_m$ и $\overline{N}_{bm}$ совпадают.

В общем случае, величины λ_m определяются следующим соотношением:

$$\lambda_m = \sum_{i_1=0}^{NK} \ldots \sum_{i_m=1}^{NK} \ldots \sum_{i_M=0}^{NK} P(i_1, \ldots i_M).$$

Тогда, в окончательном виде среднее время выполнения запроса от *m*-того процессора определяется в виде:

$$T_m = (1-R_m) + R_m \cdot \left(\frac{\overline{N}_{bm} + \overline{N}_m}{\lambda_m} \right).$$

Среднее время выполнения транзакции к подсистеме памяти от любого процессора есть усредненное на вероятность промаха в кэш время выполнения транзакции от *m*-того процессора:

$$\overline{T} = \frac{\sum_{m=0}^{M} R_m \cdot T_m}{\sum_{m=0}^{M} R_m}.$$

4.3. Пропускная способность

Операционная характеристика «Пропускная способность подсистемы памяти от *m*-того процессора» показывает, сколько транзакций доступа к памяти от *m*-того процессора выполняется в единицу времени. Она дает интегральную характеристику производительности подсистемы памяти (скорость работы).

В Марковской системе массового обслуживания, интенсивность входящего потока равна интенсивности исходящего потока [19]. Напомним, что система принимает транзакции от *m*-того ЦП, если кэш *m*-того процессора не является заблокированным $(1 - Q_m)$.

Тогда пропускная способность подсистемы памяти от *m*-того процессора задается выражением:

$$C_m = \frac{(1 - Q_m)}{t}.$$

Пропускная способность подсистемы памяти в целом определяется как сумма пропускных способностей от каждого процессора:

$$\overline{C} = \sum_{m=0}^{M} C_m .$$

5. ВЛИЯНИЕ ПАРАМЕТРОВ ПОДСИСТЕМЫ ПАМЯТИ НА ЕЕ ПРОИЗВОДИТЕЛЬНОСТЬ

Выясним влияние различных параметров подсистемы памяти на ее операционные характеристики.

5.1. Вероятность промаха в кэш

Эмпирически, при прочих равных условиях, увеличение вероятности промаха в кэш должно приводить к увеличению средней задержки. Это связано с тем, что при промахе в кэш время обработки транзакции увеличивается с t как минимум до $K \cdot t$. Также интуитивно понятно, что увеличение вероятности промаха в кэш должно приводить к уменьшению пропускной способности (т.к. вероятность блокировки в этом случае увеличивается). Проверим эти утверждения с помощью наших моделей.

На рис. 9, 10, 11 иллюстрируется влияние вероятности промаха в кэш первого процессора R_1 на среднюю задержку обработки транзакции от первого процессора T_1 при различных значениях *K*, *N*, *M* соответственно.

Из этих рисунков видно, что для любых значений параметров *K*, *M*, *N* минимальная средняя задержка $T_1 = t$ достигается в случае вероятности промаха в кэш $R_1 = 0$. В случае кэша блокирующего типа максимальная средняя задержка однопроцессорной системы (*M*=1) составляет $T_1 = K \cdot t$ при вероятности промаха $R_1 = 1$ (рис. 9). Рис. 10 показывает, что кэш неблокирующего типа $(N = 2)$ имеет меньшую среднюю задержку, чем кэш блокирующего типа $(N = 1)$ на всем диапазоне возможных значений $\vec{R}$. Однако КНТ с глубиной неблокируемости $N > 2$ имеют значительно большую среднюю задержку при достаточно больших вероятностях промаха в кэш $\vec{R}$.

На рис. 12, 13, 14 иллюстрируется влияние вероятности промаха в кэш первого процессора R_1 на пропускную способность обработки транзакций от первого процессора C_1 при различных значениях параметров *K*, *N*, *M* соответственно.

Из этих рисунков видно, что для любых значений параметров *K*, *M*, *N* максимальная пропускная способность $C_1 = \frac{1}{t}$ достигается в случае вероятности промаха в кэш $R_1 = 0$. В случае кэша блокирующего типа минимальная пропускная

способность однопроцессорной системы (M=1) составляет $C_1 = \frac{1}{(K+1)\cdot t}$ (рис. 12). Рис. 13 показывает, что пропускная способность подсистемы памяти с кэшем неблокирующего типа, всегда выше пропускной способности подсистемы памяти с кэшем блокирующего типа.

5.2. Время доступа к ОЗУ

Эмпирически, при прочих равных условиях, чем меньше время доступа к ОЗУ K, тем эффективней должна работать подсистема памяти в целом. Проверим это утверждение с помощью наших моделей.

На рис. 15, 16 иллюстрируется влияние параметра K на среднюю задержку обработки транзакции от первого процессора T_1 при различных значениях параметров N, M соответственно. Видно, что увеличение K действительно увеличивает среднюю задержку, причем увеличение средней задержки в случае многопроцессорной системы с КБТ происходит практически линейно (рис. 16), а в случае однопроцессорной системы с КНТ - не линейно (рис. 15).

На рис. 17, 18 иллюстрируется влияние параметра K на пропускную способность обработки транзакций от первого процессора C_1 при различных значениях параметров N, M соответственно. Видно, что с увеличением параметра K пропускная способность падает.

5.3. Глубина неблокируемости

Из литературы известно, что кэш неблокирующего типа используется для повышения производительности подсистемы памяти [7]. Действительно ли подсистема памяти с более высоким коэффициентом неблокируемости демонстрирует более высокую производительность?

На рис. 19, 20 иллюстрируется влияние параметра N на среднюю задержку обработки транзакции от первого процессора T_1 при различных значениях параметров K, M. Видно, что некоторое увеличение глубины неблокируемости N сначала приводит к уменьшению средней задержки, однако дальнейшее увеличение N приводит к ее увеличению.

На рис. 21, 22 иллюстрируется влияние параметра N на пропускную способность обработки транзакций от первого процессора C_1 при различных значениях параметров *K*, *M*. Видно, что при увеличении глубины неблокируемости *N* пропускная способность C_1 растет. Видно, что увеличение глубины неблокируемости приводит к увеличению пропускной способности. Однако при достаточно больших значениях *N* увеличение пропускной способности становится практически незаметным.

5.4. Количество процессоров

В настоящее время на рынке представлено множество многопроцессорных систем с общей разделяемой памятью. Число процессоров в таких системах колеблется от двух до восьми. Очевидно, что при увеличении числа процессоров, нагрузка на общую ОЗУ возрастает и производительность подсистемы памяти будет падать.

На рис. 23, 24 иллюстрируется влияние параметра M на среднюю задержку обработки транзакции от первого процессора T_1 при различных значениях параметров *K*, *N* соответственно. Видно, что увеличение количества процессоров приводит к увеличению средней задержки.

На рис. 25, 26 иллюстрируется влияние параметра M на пропускную способность обработки транзакций от первого процессора C_1 для различных значений параметров K, N соответственно. Видно, что увеличение количества процессоров приводит к некоторому уменьшению пропускной способности.

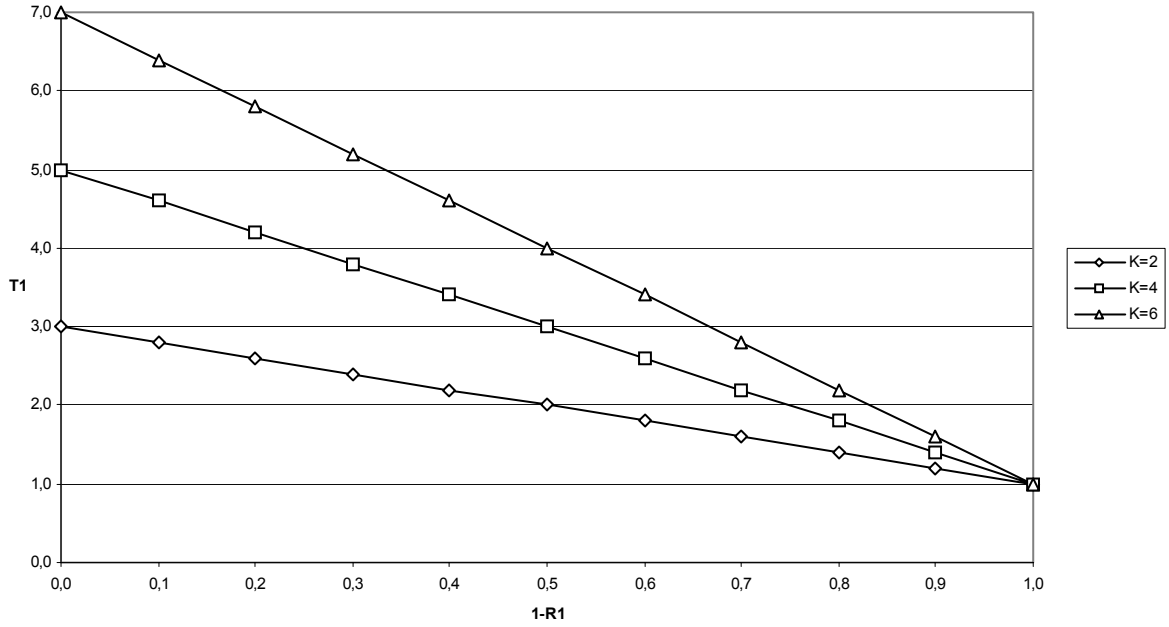

Рис. 9. Влияние вероятности промаха в кэш R_1 на среднюю задержку T_1 при различных значениях параметра K (M=1, N=1)

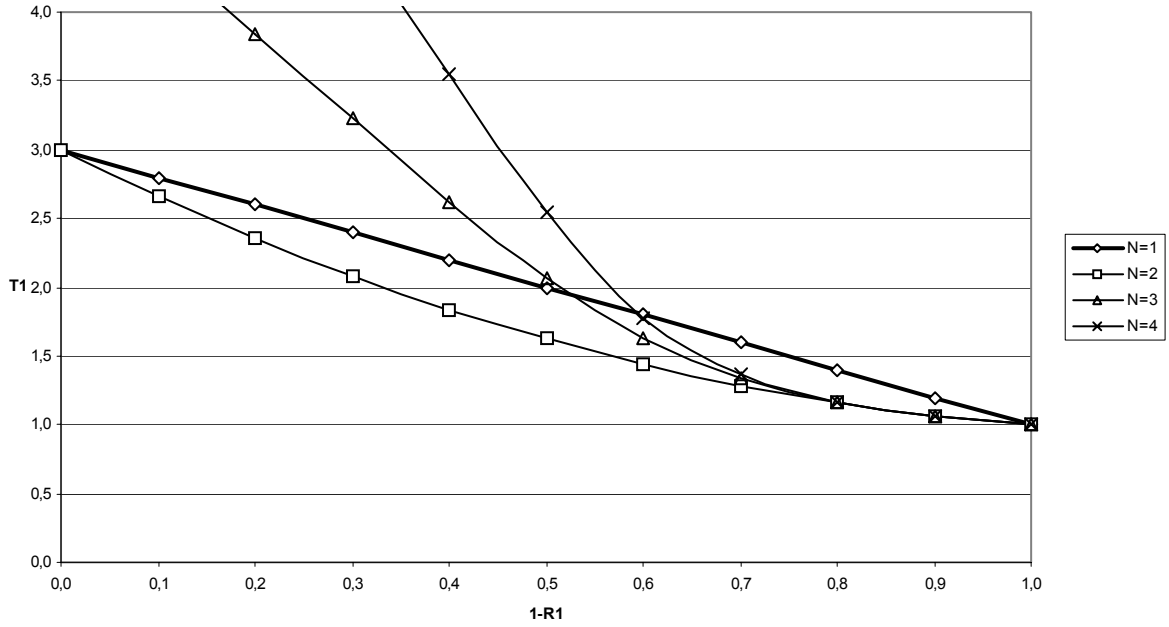

Рис. 10. Влияние вероятности промаха в кэш R_1 на среднюю задержку T_1 при различных значениях коэффициента неблокируемости N (M=1, K=2)

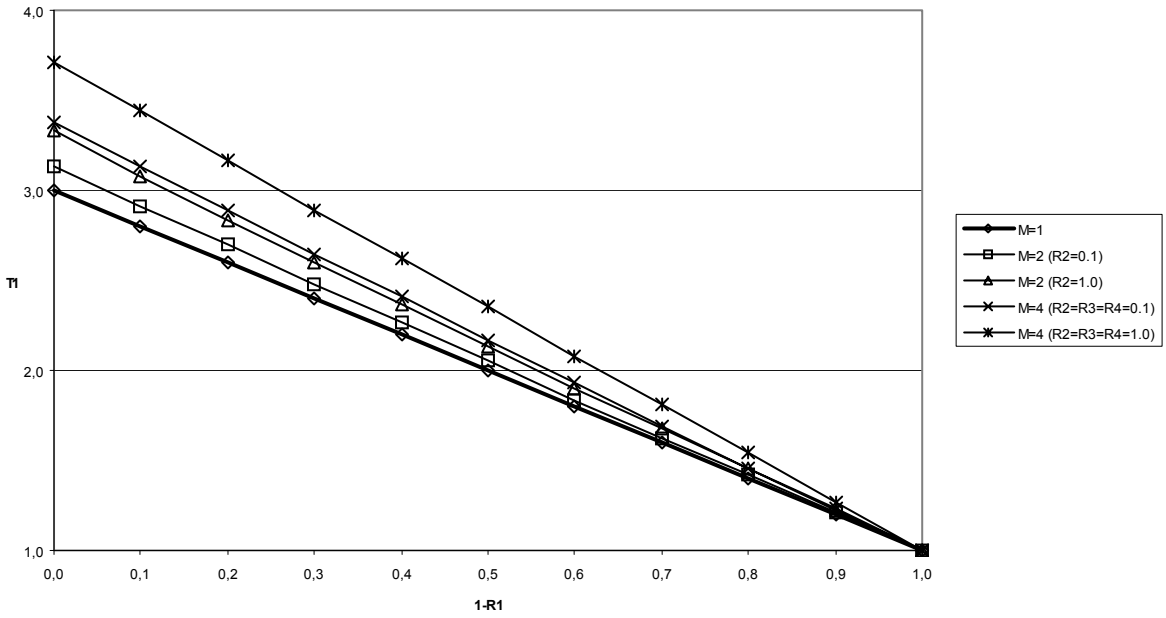

Рис. 11. Влияние вероятности промаха в кэш R_1 на среднюю задержку T_1 при различном количестве процессоров M (N=1, K=2)

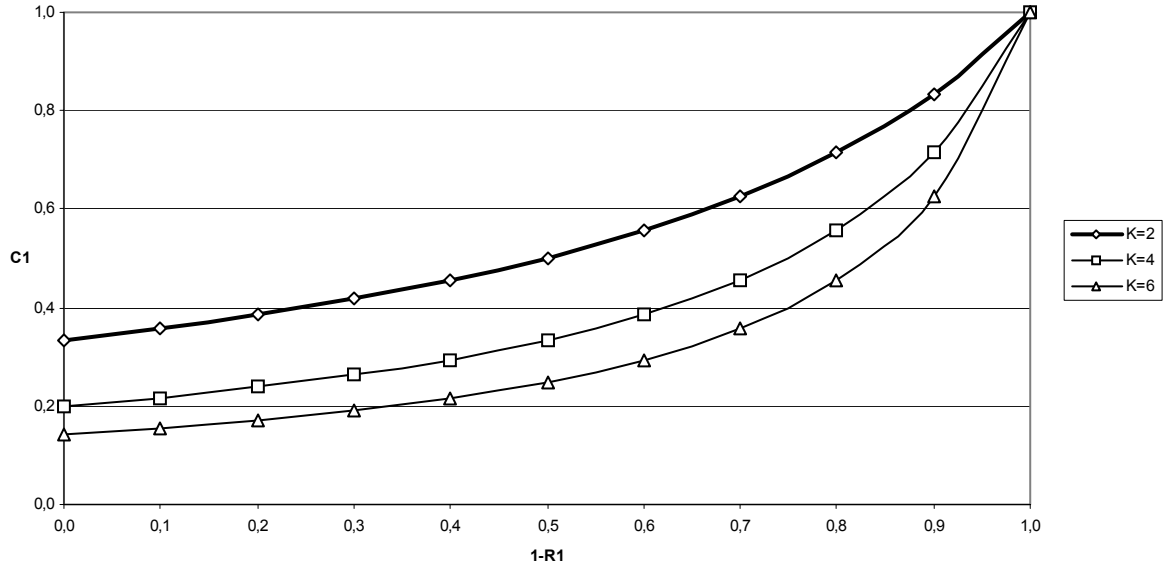

Рис. 12. Влияние вероятности промаха в кэш R_1 на пропускную способность C_1 при различных значениях параметра K (M=1, N=1)

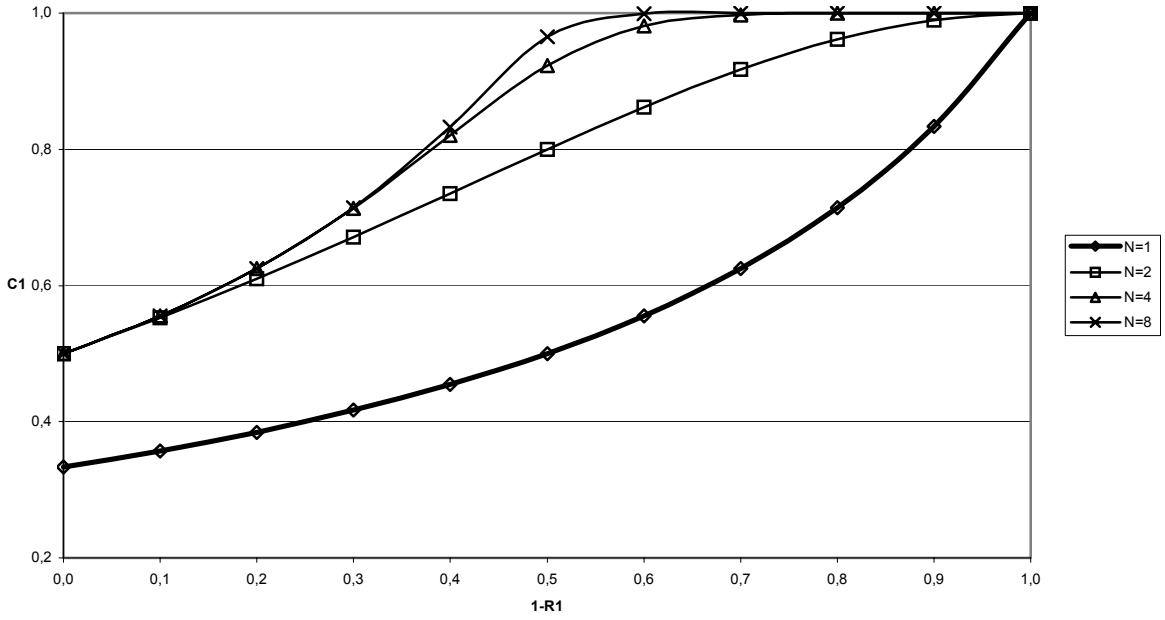

Рис. 13. Влияние вероятности промаха в кэш R_1 на пропускную способность C_1 при различных значениях коэффициента неблокируемости N (M=1, K=2)

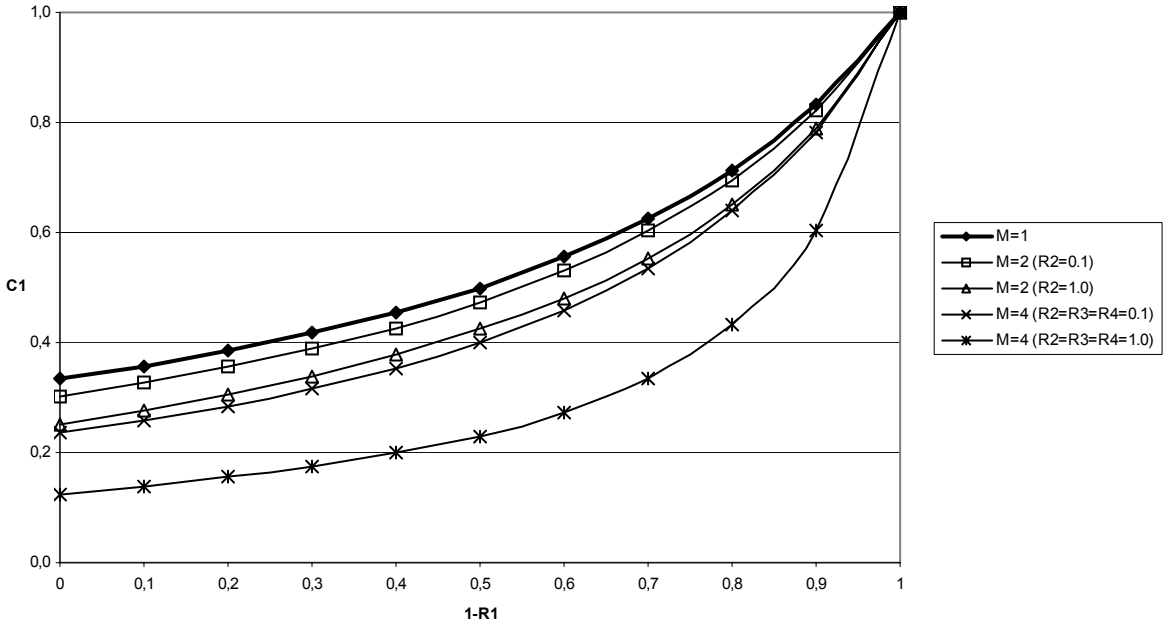

Рис. 14. Влияние вероятности промаха в кэш R_1 на пропускную способность C_1 при различном количестве процессоров M (N=1, K=2)

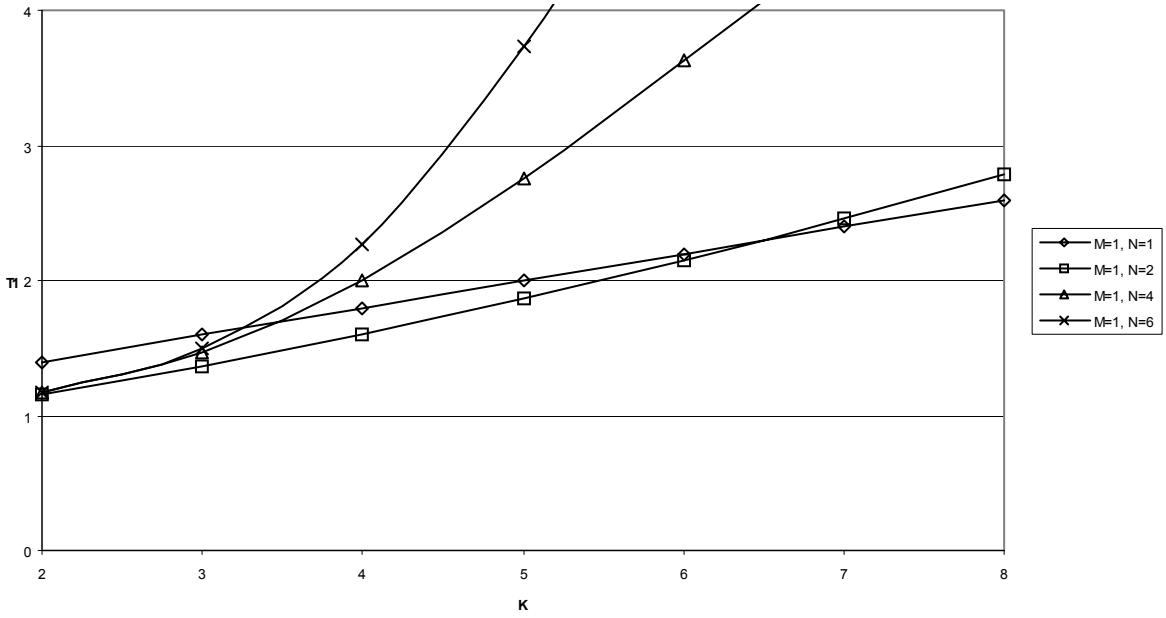

Рис. 15. Влияние параметра K на среднюю задержку T_1 при различных значениях коэффициента неблокируемости N (M=1, R=0.2)

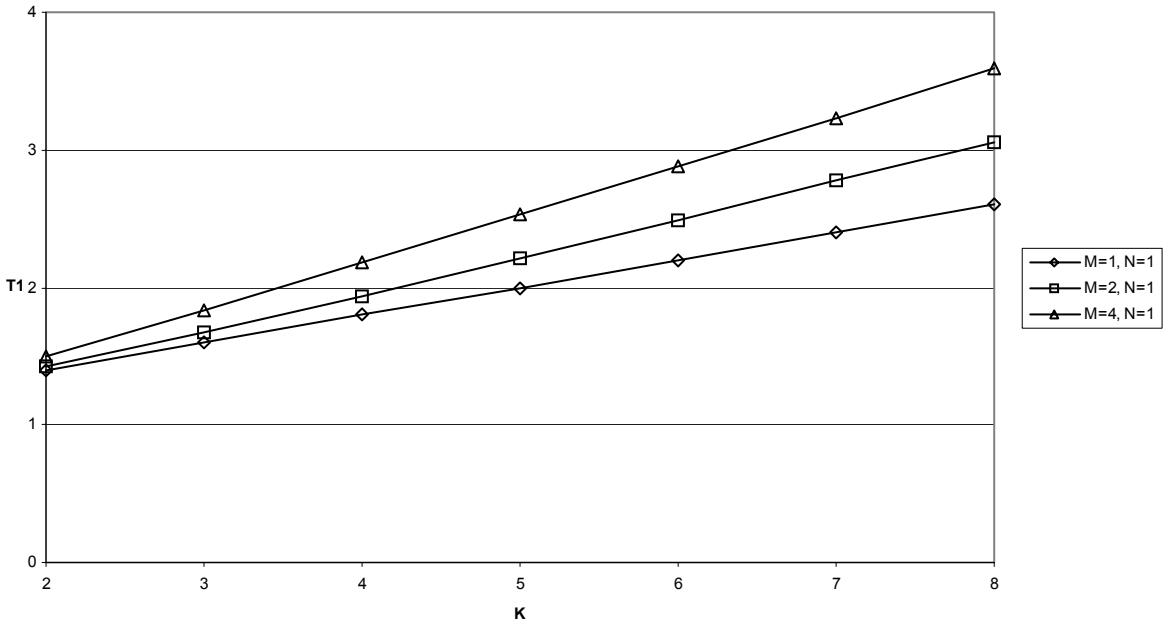

Рис. 16. Влияние параметра K на среднюю задержку T_1 при различных значениях параметра M (N=1, R=0.2)

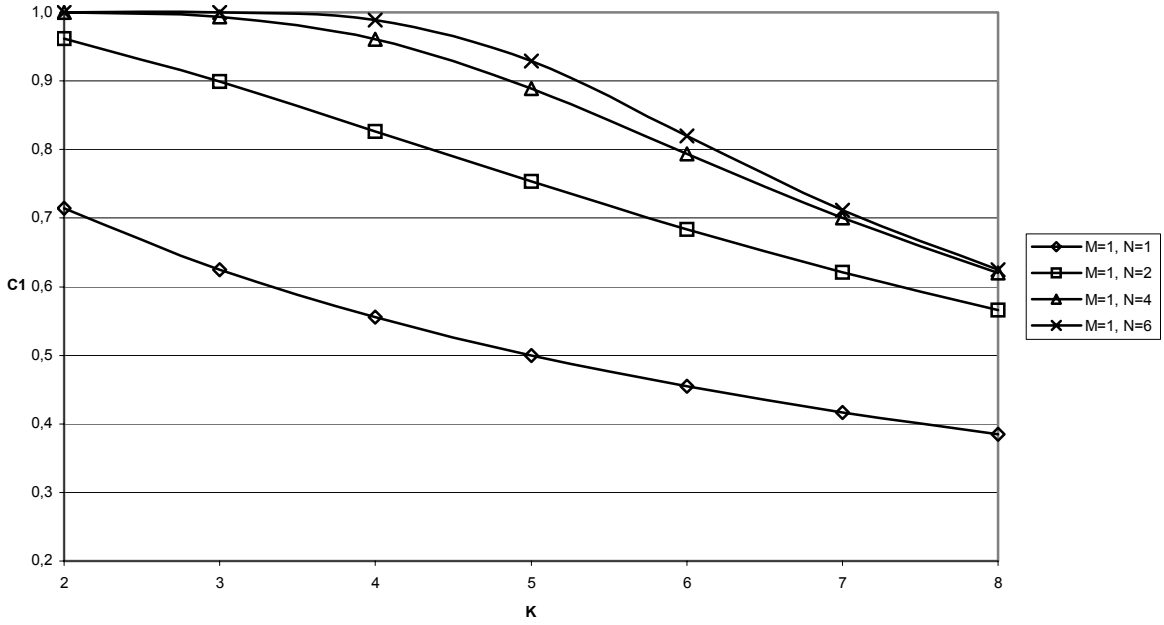

Рис. 17. Влияние параметра *K* на пропускную способность C_1 при различных значениях параметра *N* (M=1, R=0.2)

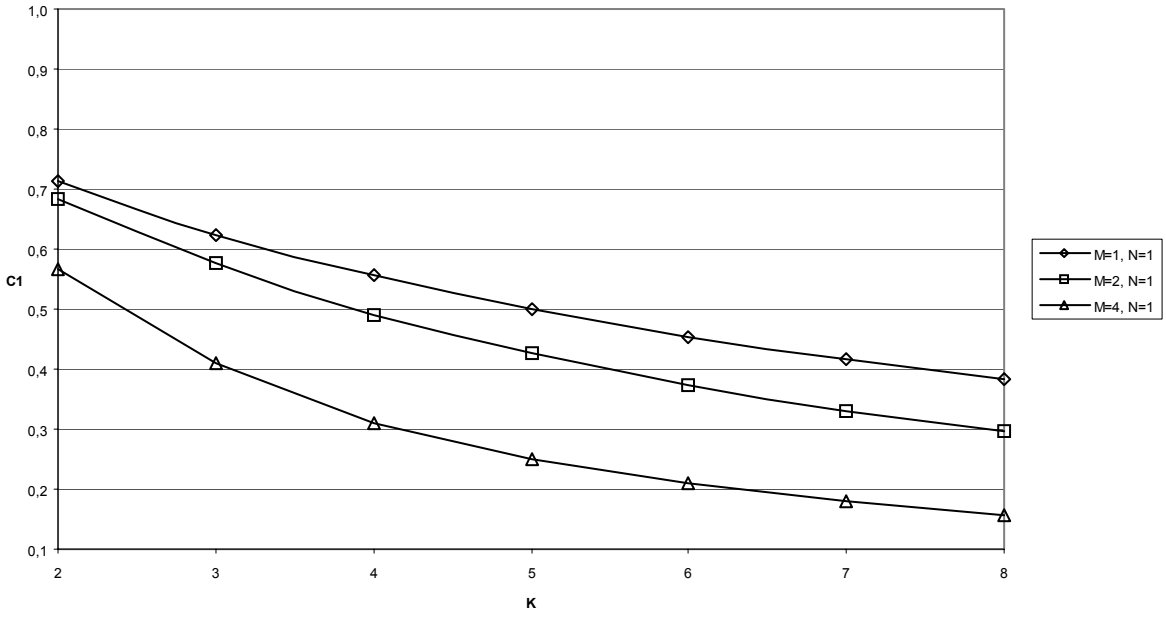

Рис. 18. Влияние параметра *K* на пропускную способность C_1 при различных значениях параметра *M* (N=1, R=0.2)

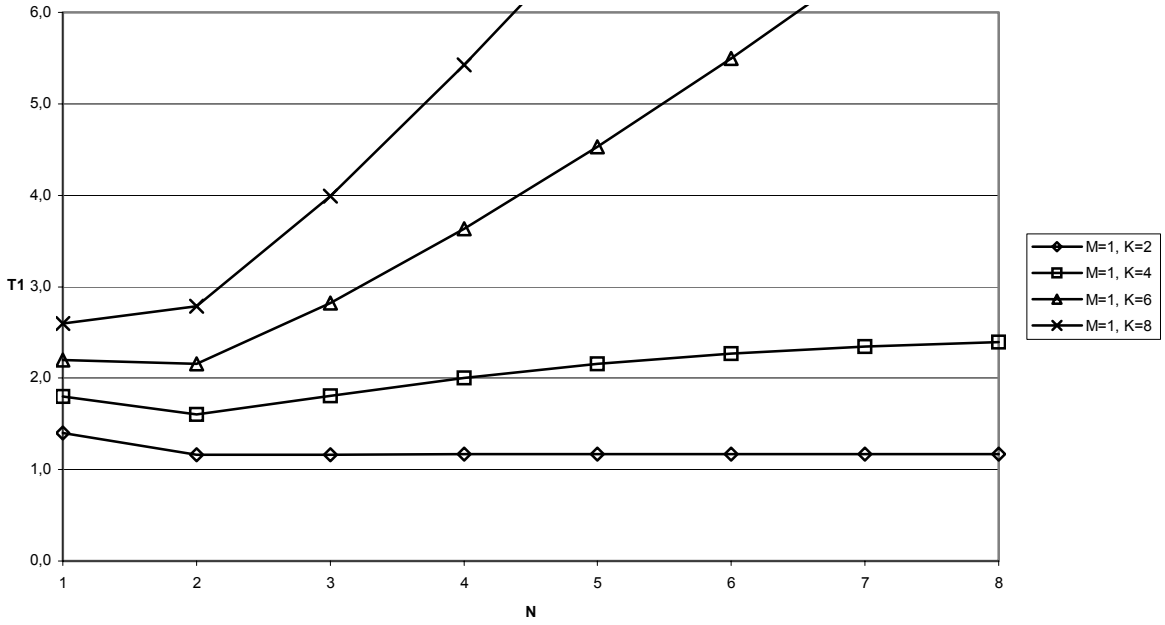

Рис. 19. Влияние глубины неблокируемости N на среднюю задержку T_1 при различных значениях параметра K (M=1, R=0.2)

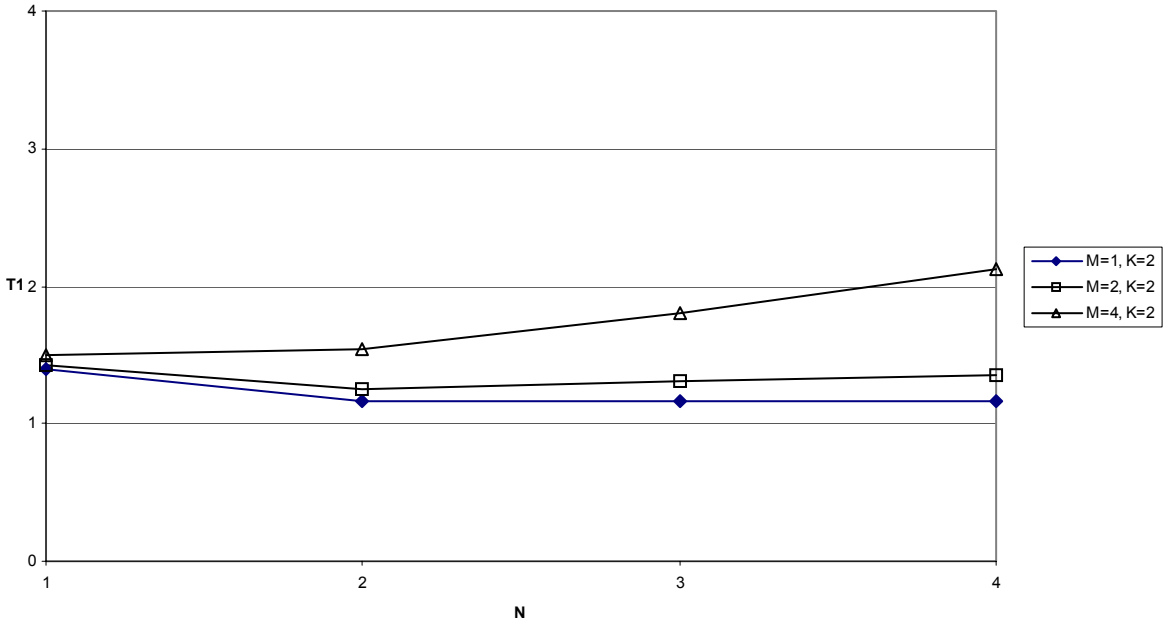

Рис. 20. Влияние глубины неблокируемости N на среднюю задержку T_1 при различных значениях параметра M (K=2, R=0.2)

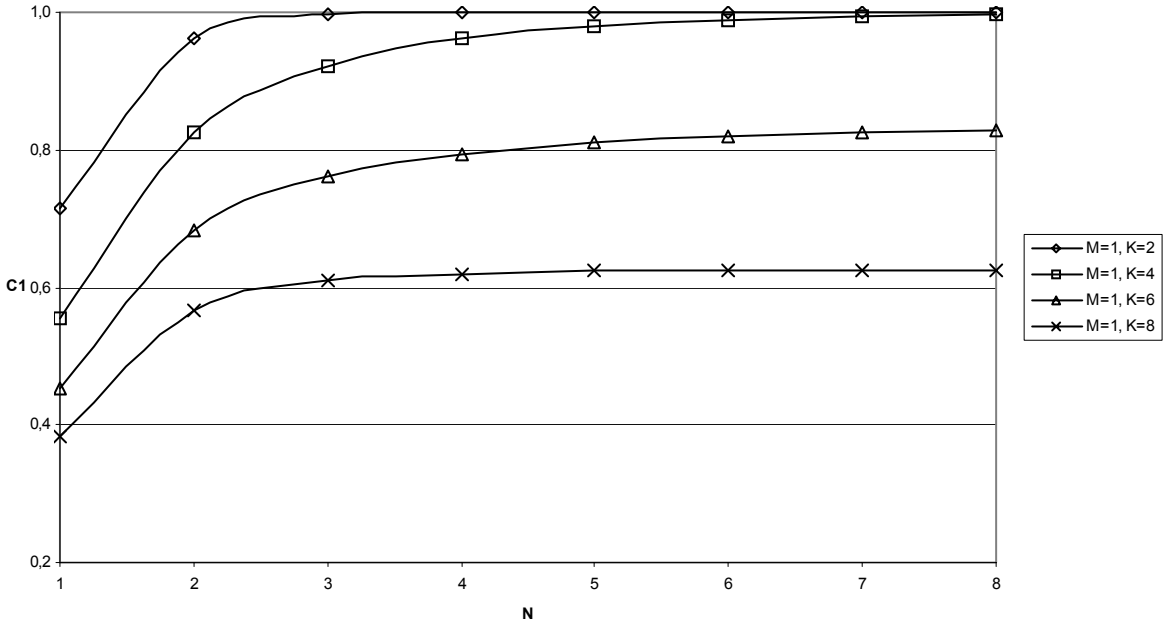

Рис. 21. Влияние глубины неблокируемости N на пропускную способность C_1 при различных значениях параметра K (M=1, R=0.2)

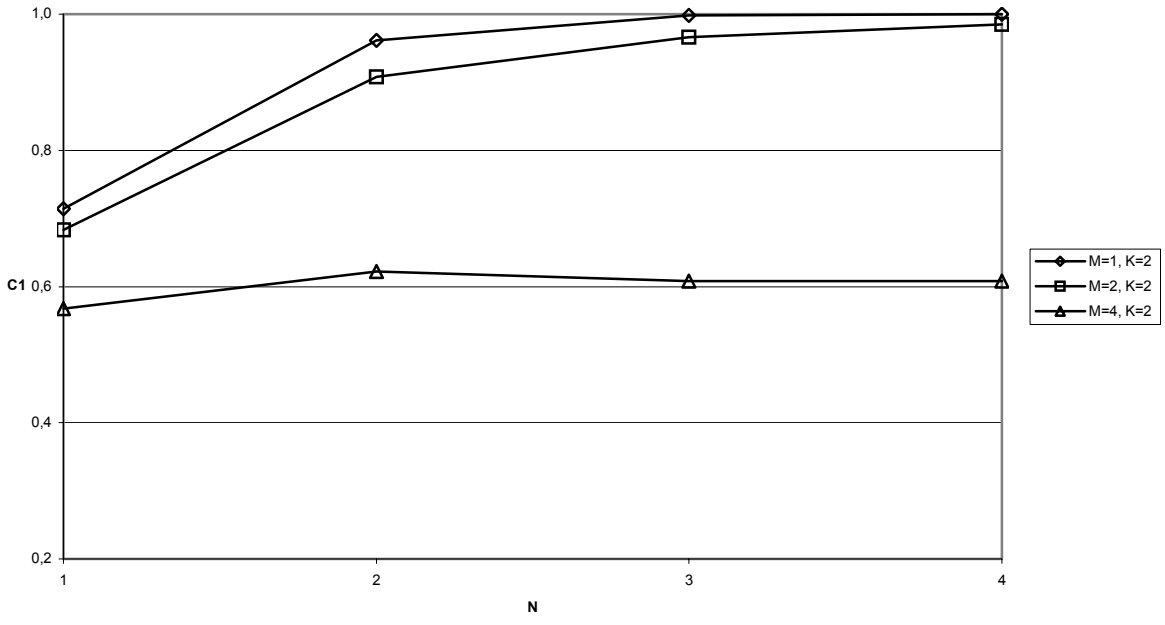

Рис. 22. Влияние глубины неблокируемости N на пропускную способность C_1 при различных значениях параметра M (K=2, R=0.2)

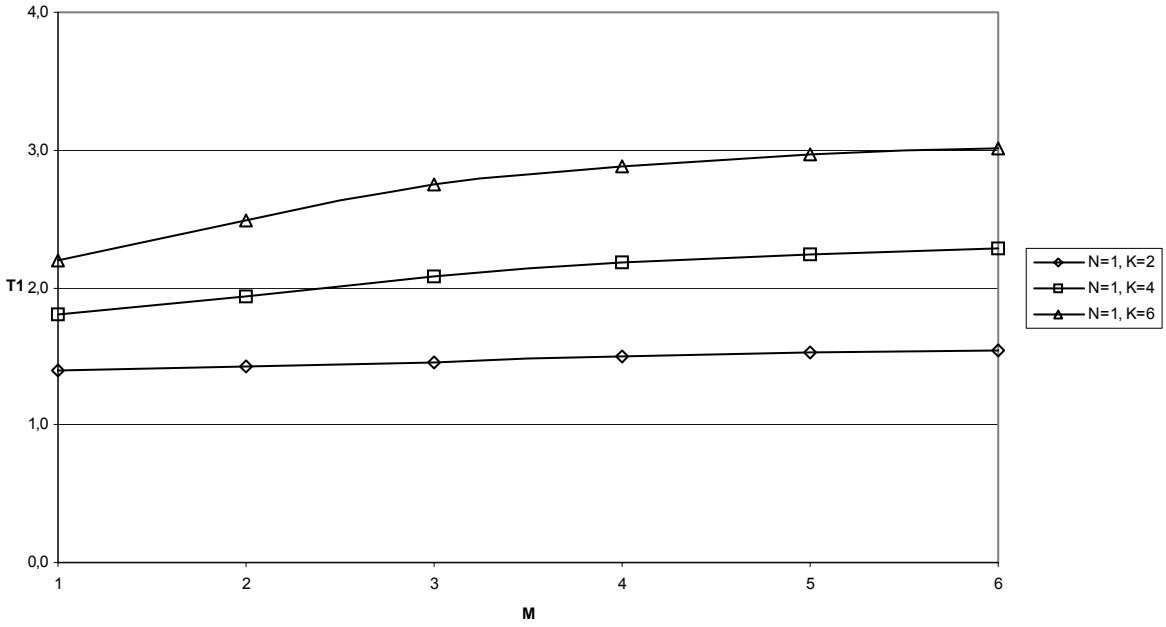

Рис. 23. Влияние числа процессоров M на среднюю задержку T_1 при различных значениях параметра K (N=1, R=0.2)

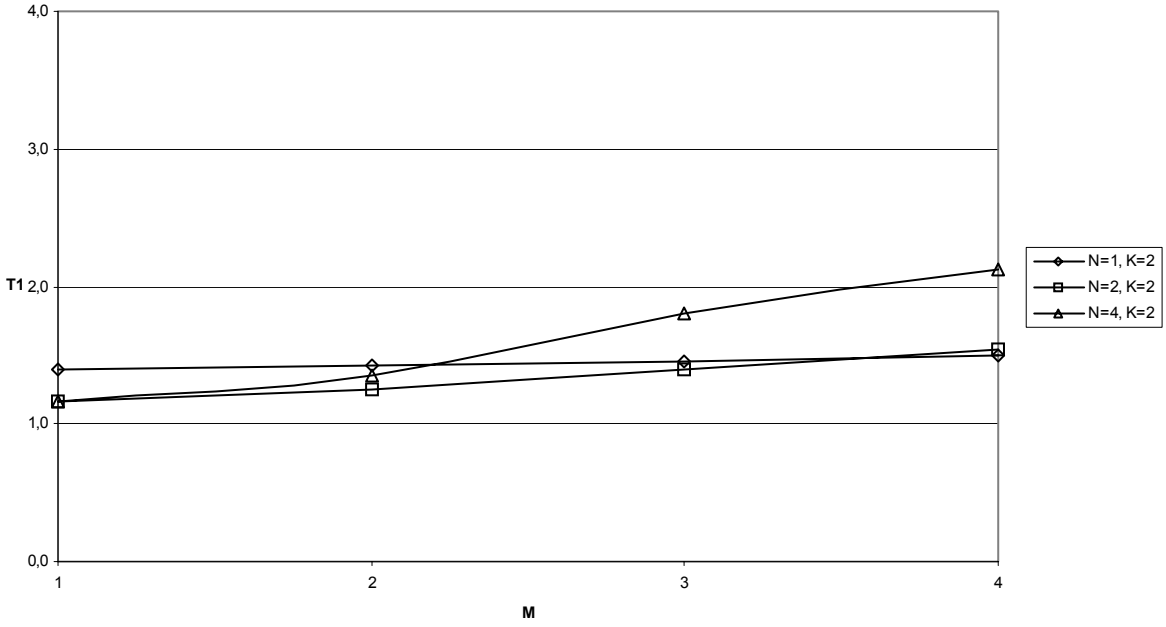

Рис. 24. Влияние числа процессоров M на среднюю задержку T_1 при различных значениях параметра N (K=2, R=0.2)

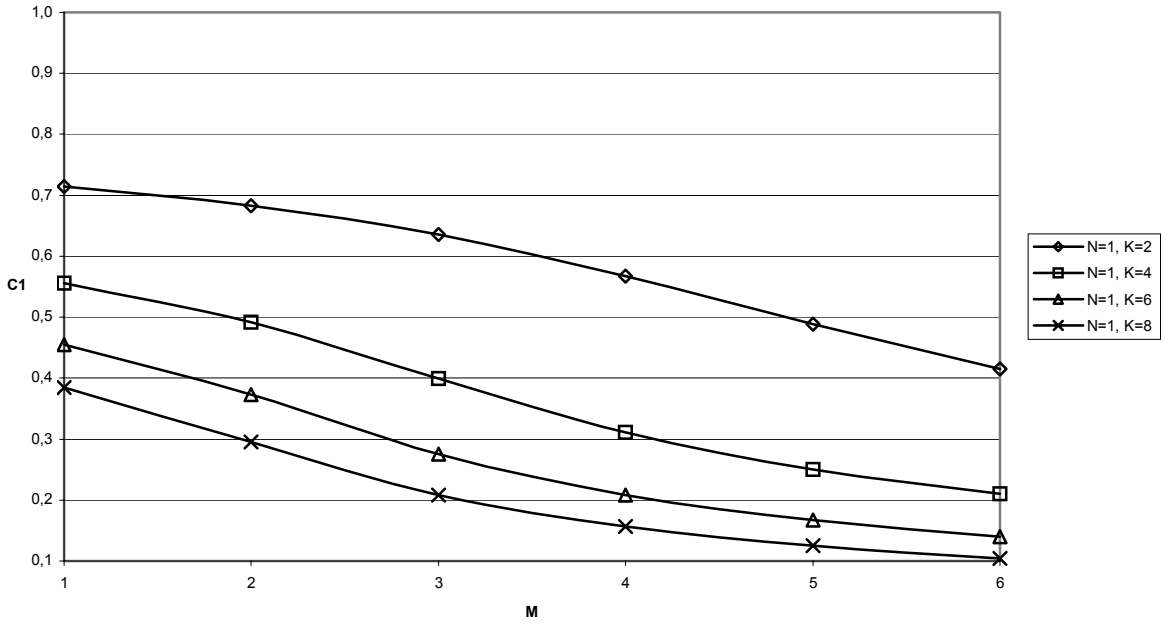

Рис. 25. Влияние числа процессоров M на пропускную способность C_1 при различных значениях параметра K (N=1, R=0.2)

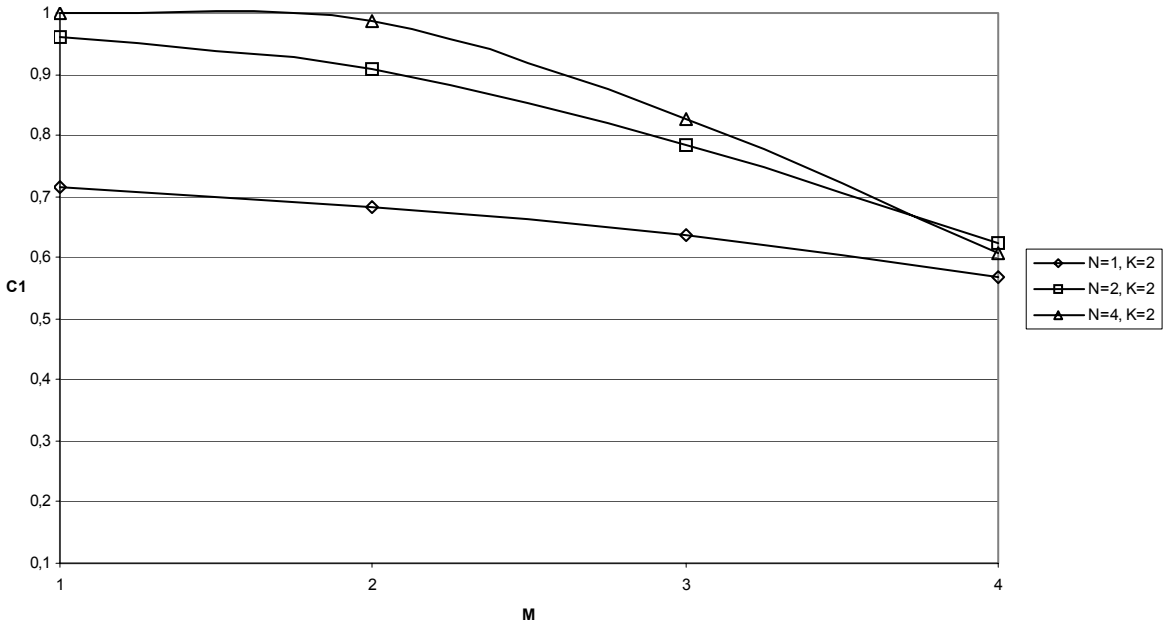

Рис. 26. Влияние числа процессоров M на пропускную способность C_1 при различных значениях параметра N (K=2, R=0.2)

ЗАКЛЮЧЕНИЕ

В результате проведенного исследования были изучены различные архитектуры построения многоуровневой памяти для многопроцессорных SMP-систем. Выявлены и унифицированы их общие свойства. На основе этих знаний были построены подробные и адекватные математические модели функционирования двухуровневой подсистемы памяти неблокирующего типа для многопроцессорной системы.

Для однопроцессорных подсистем памяти были получены аналитические зависимости для нахождения вероятностей состояний в широком спектре значений параметров N, K, $\vec{R}$. Для систем с двумя и большим количеством процессоров получены аналитические зависимости для нахождения операционных характеристик при определенных, наиболее показательных наборах параметров N, K, $\vec{R}$.

Разработанное приложение «SMPMemMark» позволяет численно находить вероятности состояний и операционные характеристики для подсистем памяти с любыми значениями параметров M, N, K, $\vec{R}$.

Численный анализ влияния параметров подсистемы памяти выявил следующие пути увеличения производительности подсистем памяти:

- уменьшение вероятности промаха в кэш $\vec{R}$;
- уменьшение времени обращения к кэшу t;
- уменьшение времени доступа к оперативной памяти $K \cdot t$.

Увеличение глубины неблокируемости N в некотором диапазоне значений параметров $\vec{R}$, K, M позволяет увеличивать ее производительность. Чем больше глубина неблокируемости, тем уже этот эффективный диапазон параметров. Увеличение количества ЦП M в некотором диапазоне параметров $\vec{R}$, K, M незначительно снижает производительность подсистемы памяти. Чем больше количество процессоров, тем больше падение производительности при дальнейшем увеличении количества ЦП.

Развитие данного исследования возможно в следующих направлениях:

- исследование многоуровневой подсистемы памяти неблокирующего типа и оценка ее производительности;
- анализ эффективности функционирования различных механизмов обеспечения когерентности кэшей в многопроцессорных системах.

ЛИТЕРАТУРА

1. Хамахер К., Вранешич З., Заки С. Организация ЭВМ. – СПб: Питер, 2002.
2. Таненбаум Э. Архитектура компьютера. – СПб: Питер, 2002.
3. Корнеев В.В., Киселев А.В. Современные микропроцессоры. – М.: Нолидж, 1998. – 240с.
4. Лускинд Ю.И. Буферные запоминающие устройства типа кэш //Зарубежная радиоэлектроника, 1990, №4. С. 29-42.
5. Феррари Д. Оценка производительности вычислительных систем. – М.: Мир, 1981.
6. Юрченко А.С. К выбору конфигурации иерархической системы памяти //Автоматика и телемеханика, 1985, №4. С. 137-139.
7. Шнитман В.З. Современные высокопроизводительные компьютеры //Аналитические материалы Центра информационных технологий. http://www.citforum.ru.
8. Шнитман В.З., С.Д. Кузнецов Аппаратно-программные платформы корпоративных информационных систем //Аналитические материалы Центра информационных технологий. http://www.citforum.ru.
9. Пом А., Агравал О. Быстродействующие системы памяти. – М.: Мир, 1987.
10. Биматов Д.В., Севостьянов Д.В., Сущенко М.С., Сущенко С.П. Вероятностные модели кэша: анализ эффективности //Сборник трудов V Всероссийской конференции «Наука и образование». – Томск: Изд-во ТГПУ, 2001.
11. Биматов Д.В., Севостьянов Д.В, Сущенко С.П. Анализ эффективности процессорного кэша //Тезисы докладов 2-ой Международной конференции молодых ученых и аспирантов «Актуальные проблемы современной науки», Самара: Изд-во СамГТУ, 2001.
12. Биматов Д.В., Сущенко С.П. Исследование эффективности кэша неблокирующего типа //Материалы XLI Международной научной студенческой конференции «Студент и научно-технический прогресс», – Новосибирск: Изд-во НГУ, 2003.
13. Кохонен Т. Ассоциативные запоминающие устройства: Пер. с англ. – М.: Мир, 1982. – 384с.
14. Сущенко М.С., Сущенко С.П. Анализ эффективности многоуровневой памяти вычислительных систем //Обозрение прикл. и промышл. матем., 2001, т. 8, в. 1, с. 336-337.

15. Сущенко М.С., Сущенко С.П. Модель влияния глубины неблокируемости кэша на быстродействие многоуровневой памяти //Обозрение прикл. и промышл. матем., 2001, т. 8, в. 2, с. 695-696.
16. Биматов Д.В., Сущенко М.С., Сущенко С.П. Моделирование разделяемой памяти двухпроцессорной вычислительной системы //Вестник Томского государственного университета. – Томск: Изд-во ТГУ, 2003 (в печати).
17. Феллер В. Введение в теорию вероятностей и ее приложения. – М.: Мир, 1984. – Т.1.
18. Теория проектирования вычислительных машин, систем и сетей /Под ред. засл. деят. науки РФ, д-ра техн. наук, проф. В.И. Матова. – М.: МАИ, 1998. – 460 с.
19. Клейнрок Л. Теория массового обслуживания. – М.: Машиностроение, 1979.
20. Хэмминг Р.В. Численные методы. – М.: Наука, 1972.

ПРИЛОЖЕНИЕ А. РУКОВОДСТВО ПРОГРАММИСТА

В рамках дипломной работы было разработано приложение «SMPMemMark», позволяющее моделировать подсистему памяти с заданными параметрами M, N, K, $\vec{R}$.

Приложение разработано в среде Microsoft Visual Studio .NET на языке C++. Благодаря тому, что приложение использует только стандартные библиотеки, оно может быть с незначительными изменениями перекомпилировано на произвольной платформе с помощью компилятора, поддерживающего стандарт ANSI C/C++.

Структура приложения выглядит следующим образом:

- функция Enter() – ввод исходных данных;
- функция Solve() – вычисление вероятностей состояний Марковской СМО и ее операционных характеристик;
- функция Out() – вывод результатов пользователю.

Все вышеперечисленные функции возвращают флаг успешности выполнения операции.

Стоит отметить, что ввод данных с клавиатуры часто оказывается неудобным для пользователя. Для построения графиков требуется пакетное выполнение программы. Для этих целей была добавлена отдельная функция Start(), которая выполняет одну итерацию работы приложения по вводу данных, расчету операционных характеристик и выводу их в стандартный поток вывода.

Функция Solve() выполняет следующие действия:

- генерирует все корректные состояния соответствующей цепи Маркова.
- генерирует матрицу переходов соответствующей цепи Маркова;
- итерационно вычисляет вероятности состояний соответствующей цепи Маркова;
- вычисляет операционные характеристики подсистемы памяти на основе вероятностей состояний.

Корректные состояния цепи Маркова генерируются следующим образом: для каждого потенциально возможного состояния системы определяется, является ли оно корректным. Трудоемкость генерации корректных состояний составляет $T_{states} = M \cdot (NK+1)^M$ элементарных операций.

Матрица переходов цепи Маркова генерируется следующим образом: для каждого состояния системы определяются возможные переходы в другие состояния и вероятности этих переходов по формуле (2.2). Трудоемкость генерации матрицы переходов составляет $T_{matrix} = nStates \cdot nJumps$, где

$nStates = (N+1)^M + (N+1)^{M-1} \cdot M \cdot N \cdot (K-1)$ - количество корректных состояний (2.1), а $nJumps = M \cdot 2^M$ - максимально возможное количество переходов из состояния (2.3).

Вероятности состояний вычисляются следующим образом. Произвольно выбирается начальный вектор вероятностей состояний. Затем текущий вектор вероятностей состояний многократно умножается на матрицу переходов (3.10). Процесс завершается, когда вектор состояний $\pi^{(n)}$, полученный на последней итерации, практически не отличается от вектора, полученного на предыдущей итерации $\pi^{(n-1)}$:

$$\max_{j=1,M} \left| \pi^{(n)} - \pi^{(n-1)} \right| < \varepsilon,$$

где $\varepsilon > 0$ – требуемая точность вычислений. Трудоемкость вычисления вероятностей состояний составляет $T_{solve} = nStates \cdot n$ элементарных операций, где n - количество итераций, зависящее от ε.

Трудоемкость вычисления операционных характеристик составляет $T_{tth} = nStates$.

Общая трудоемкость составляет $T = O(T_{states} + T_{matrix} + T_{solve} + T_{tth}) = O\left(2^M \cdot K \cdot N^M\right)$.

Понятно, что моделирование подсистем памяти с большим количеством процессоров и (или) большим коэффициентом неблокируемости требует значительных вычислительных ресурсов. К тому же, требуется хранить в оперативной памяти матрицу из $nStates \cdot nStates = O\left(N^{2M}\right)$ элементов.

Приложение активно использует написанную автором библиотеку классов матрицы и вектора (CIntVector, CDoubleVector, CDoubleMatrix). Расчет и хранение параметров Марковской цепи локализованы в разработанных автором классах Марковской СМО (CMarkSystem) и состояния Марковской СМО (CState). Ниже приведены прототипы этих классов.

```
// ============================================================================
// CMarkSystem - класс для марковской СМО
// используется как контейнер параметров марковской СМО, состояний и т.д.
// ============================================================================

// константы для режимов системы
const
      MS_BEFORE_INIT                =     0x0001,
      MS_BEFORE_CALCEQUATIONS       =     0x0010,
      MS_BEFORE_CALCTTH             =     0x0100,
      MS_AFTER_CALCTTH              =     0x1000;

class CMarkSystem
{
// данные
private:
      int m_Status;                 // текущий режим системы

// параметры системы
      int m_ParamM;                 // количество ЦП
      int m_ParamN;                 // глубина неблокируемости
      int m_ParamK;                 // время доступа к ОЗУ (в тактах кэша)
      CDoubleVector m_ParamR;       // вероятности промаха в кэши

// вычисляемые данные
      int m_MaxStateDigit;          // максимальная цифра в номере состояния
      int m_MaxFreeDigit;           // последняя цифра в номере состояния
      int m_StateCount;             // число состояний в системе
      int m_ValidStateCount;        // число корректных состояний в системе
      CState **m_pStates;           // массив состояний системы

// матрица переходов (заполняется на основе состояний...)
      CMarkMatrix m_Matrix;         // матрица переходов
      CDoubleVector m_ResultVector; // вектор вероятностей состояний

// вектора операционных характеристик (для каждого ЦП)
      CDoubleVector m_TTH_N;        // число этапов в системе
      CDoubleVector m_TTH_Nb;       // число этапов до разблокировки кэша
      CDoubleVector m_TTH_S;        // вероятность обслуживания ЦП
      CDoubleVector m_TTH_Q;        // вероятность блокировки ЦП

      CDoubleVector m_TTH_T;        // средняя задержка
      CDoubleVector m_TTH_C;        // пропускная способность


// защищенные методы
protected:
      void KillAll();
      void KillStates();
      void NewStates();

      void CalcValidStateCount();
      void FillMatrix();
      bool CheckMatrix();

      bool IterateSolveEquation();
      void SetStateProbabilities();

// нахождение операционных характеристик
      bool CalcBaseTTH();
      bool CalcUserTTH();

// деструктор и конструктор
```

```
public:
	~CMarkSystem();
	CMarkSystem();

// методы
public:
	int GetParamM();
	int GetParamN();
	int GetParamK();
	double GetParamR(int index);
	CDoubleVector& GetRVector();
	int GetStatus();

	CMarkMatrix& GetMatrix();
	int GetMaxStateDigit();
	int GetMaxFreeDigit();
	int GetStateCount();
	int GetValidStateCount();


// доступ к операционным характеристикам
	double GetTTH_N(int index);
	double GetTTH_Nb(int index);
	double GetTTH_S(int index);
	double GetTTH_Q(int index);
	double GetTTH_T(int index);
	double GetTTH_C(int index);

	CDoubleVector& GetVectorTTH_N();
	CDoubleVector& GetVectorTTH_Nb();
	CDoubleVector& GetVectorTTH_S();
	CDoubleVector& GetVectorTTH_Q();
	CDoubleVector& GetVectorTTH_T();
	CDoubleVector& GetVectorTTH_C();

// получить состояние по ID или по номеру
	CState* GetState(int _ID);
	CState* GetState(CIntVector &vNumber);

	bool IncrementNumber(CIntVector &vNumber);
	int GetNumberID(CIntVector &vNumber);

// инициазировать систему параметрами
	bool Init(int ParamM, int ParamN, int ParamK, CDoubleVector &ParamR);

// найти вероятности состояний
	bool CalcEquations();

// найти операционные характеристики
	bool CalcTTH();

};
```

```
// ============================================================================
// CState - класс для состояния марковской СМО
// хранит номер, id и другие параметры состояния
// ============================================================================

class CState
{
private:
	CMarkSystem *m_pOwner;		// владелец

	int m_ID;			// идентификатор
	CIntVector m_Number;		// номер вектора
	CIntVector m_FreeMask;		// маска возможных направлений движения

	CDoubleVector m_Jump;		// вектор вероятностей переходов

	double m_Probability;		// вероятность состояния

// защищенные методы
protected:
	void CalcFreeMask();

	int GetRollbackCount();
	bool GetRollback(CIntVector &vRollback, int index);
	bool GetNextRun(CIntVector &vRun);
	double GetRunProbability(CIntVector &vRun, double InitFactor);
	void TransformNumber(CIntVector &vRollback, CIntVector &vRun); int
	GetNewNumberID(CIntVector &vRollback, CIntVector &vRun);

// методы
public:
	// деструктор и конструктор
	~CState();
	CState(CMarkSystem &rOwner, int _ID, CIntVector &vNumber);

	int GetID();

	CIntVector& GetNumberVector();
	int GetNumber(int index);
	CIntVector& GetFreeMask();
	CDoubleVector* GetJumpVector();
	double GetJump(int index);

	double GetProbability();
	void SetProbability(double newVal);

	bool FillJumpVector();

};
```

ПРИЛОЖЕНИЕ Б. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Приложение «SMPMemMark» функционирует под управлением следующих операционных систем:

- Microsoft Windows 9.x/ME;
- Microsoft Windows NT/2000/XP.

Для работы приложения под управлением других операционных систем требуется незначительная модификация исходного кода и перекомпиляция с помощью доступного компилятора, поддерживающего стандарт ANSI C/C++.

Приложение занимает не более 250Кбайт свободного дискового пространства. Рекомендуется запускать приложение на системе с более чем 64Мб ОЗУ.

Входные параметры:

- Количество процессоров *M*;
- Глубина неблокируемости кэшей *N*;
- Время выбора адресуемого элемента из ОЗУ *K* (в тактах кэша);
- Вектор вероятностей промаха $\vec{R} = (R_1, ... R_M)$.

Вывод:

- Количество состояний соответствующей цепи Маркова (*nStates*);
- Матрица переходов соответствующей цепи Маркова (размерность матрицы составляет *nStates***nStates* элементов);
- Вектор вероятностей состояний (*nStates* элементов);
- Операционные характеристики цепи Маркова:
    - Среднее время выполнения транзакции доступа к памяти от каждого ЦП (вектор из *M* элементов);
    - Пропускная способность подсистемы памяти от каждого ЦП (вектор из *M* элементов).

По желанию пользователя можно ограничиться выводом только некоторых из вышеперечисленных характеристик (например, только операционных характеристик).

Возможно пакетное выполнение приложения на основе перенаправления потоков ввода и (или) вывода в соответствующие файлы на диске.