16. German Rigau, Jordi Atserias, Enekko Agirre. Combining unsupervised lexical knowledge methods for word sense disambiguation. ACM, 1996.
17. Yael Karov. Similarity-based word sense disambiguation. Computation Linguistics. – Vol. 24, No 1, 1998.
18. Shigeaki Sakurai, Akihiro Suyama. Rule discovery from textural data based on key phrase patterns. ACM, 2004.
19. Benjamin Rosenfeld, Ronen Feldman, Moshe Fresko TEG — A hybrid approach to information extraction. ACM, 2004.
20. C.J.van Rijsbergen. Information Retrieval. Buttersworth, 1979.

Статья поступила в редакцию 22.05.2007

УДК 004.942: 519.876.5

А. М. Андреев, Д. В. Березкин,
Р. С. Самарев, В. В. Сюзев

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ РАЗРАБАТЫВАЕМЫХ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ НА ИХ ОСНОВЕ С ИСПОЛЬЗОВАНИЕМ АЛГЕБРАИЧЕСКИХ МОДЕЛЕЙ

Рассмотрены аспекты моделирования производительности систем управления базами данных и информационных систем, приведены метод оценки характеристик на основе алгебры процессов и пример моделирования.

Моделирование характеристик систем управления базами данных (СУБД) и информационных систем (ИС) актуально в следующих случаях: создание новой СУБД, проверка допустимых режимов работы уже существующей СУБД и оптимизация работы существующих приложений СУБД.

Архитектура СУБД наиболее существенно влияет на производительность будущих систем, построенных с применением данной СУБД. Целесообразно моделировать работу СУБД и возможные последствия от введения каких-либо элементов архитектуры СУБД на ранних стадиях разработки. Кроме того, существует возможность оценить области применения СУБД по предельно допустимым режимам работы.

В случае, когда СУБД уже разработана, необходимо знать допустимые режимы ее эксплуатации (типы приложений и наборы операций, которые не приведут к снижению производительности), а также иметь возможность оценивать требуемые параметры оборудования, такие как

быстродействие процессоров, дисков, их число, схему объединения в кластер и пр.

Если СУБД находится в эксплуатации, уже разработаны ее приложения, а выясняется, что имеющаяся вычислительная система (ВС) не обеспечивает требуемого уровня производительности или что необходимую ВС невозможно реализовать на данном уровне развития техники, тогда нужно проектировать новую систему, либо проводить оптимизацию существующей.

Среди методов моделирования производительности ИС в целом и серверов СУБД в частности можно выделить:

методы, основанные на коммуникации клиентов и серверов данных. В работе A. Delis, R. Roussopoulos [1] модели строятся и анализируются средствами СМО. Оценка производительности ИС проводится в предположении, что поток запросов клиентов однороден, возможна оценка времени их выполнения в среднем, а также с использованием различных каналов связи и кэш-методов. Однако этот уровень абстракции не позволяет оценить влияние внутренних особенностей обработки запросов сервером СУБД;

модели хранения данных на ВЗУ, позволяющие оценить минимальное время доступа к данным и производительность ИС в целом. Могут учитываться физические характеристики накопителей данных (время позиционирования головок накопителей на жестких магнитных дисках (НЖМД), скорость считывания и схема разметки дисков) [2] или методы размещения данных на них [3]. Модели первой группы в настоящее время мало применимы, поскольку большинство характеристик накопителей различных производителей не является адекватно сравнимыми;

модели имитационного представления внутренней обработки данных, которые описывают сервер СУБД с позиции последовательности обработки запросов [2, 4] и могут учитывать транзакционную обработку, разнородность запросов, а также различные методы их выполнения. Данный класс моделей может быть реализован любыми математическими средствами;

методы на основе моделей запросов к СУБД, используемые при анализе характеристик ИС в целом, а не сервера СУДБ или тракта клиент-сервер. Существуют методы, основанные на аналитическом описании процесса обработки запросов эмпирическими формулами, исходя из априорного предположения об однотипности функционирования коммерческих СУБД и однотипности выполняемого набора операций: проекции, слияния и выборки данных [5, 6]. К этому же классу моделей можно отнести анализ исходных запросов в алгебраической форме, способов их преобразования в физический план конкретной

СУБД и выполнение при помощи модели, имитирующей внутреннюю обработку запросов конкретной СУБД [8]. Модели могут быть реализованы как средствами СМО, так и в форме алгебры процессов.

Условно в отдельную группу можно выделить *модели, в которых проводится анализ производительности сервера СУБД* на основе имеющихся данных о порядке обработки данных в транзакции применительно к конкретной ИС. Выявлены модели, в которых анализ производительности проводится в зависимости от числа элементов данных внутри транзакции [8], и модели блокировки данных применительно к транзакциям OLTP систем [9].

По результатам анализа других работ можно заключить, что методы, ориентированные на моделирование характеристик отдельных подсистем или сервера СУБД в целом, не позволяют достоверно оценивать поведение ИС на основе СУБД. Для получения достоверных характеристик ИС в целом необходимо проводить анализ логических и физических планов выполнения запросов, а также учитывать особенности выполнения микроопераций физического плана сервером СУБД и их взаимного влияния.

Полная модель ИС на основе СУБД, таким образом, включает в себя следующие частные модели: модель взаимодействия клиента и сервера СУБД, в том числе многозвенные клиент-серверные архитектуры; модель преобразователя исходных запросов в логический план выполнения по схеме данных; модель оптимизатора логического плана; модель планировщика физического плана выполнения; модели функциональных обработчиков запросов.

Для моделирования параллельной СУБД необходим метод, обеспечивающий возможность моделирования параллельных процессов с учетом расщепления процессов внутренней обработки на параллельные процессы, что обусловлено применением внутризапросных форм параллелизма [10]. Кроме того, необходима композиция отдельных частных моделей в более крупные модели. В качестве математического метода моделирования целесообразно использовать алгебраические методы, поскольку именно они позволяют получить связную модель различных уровней ИС. В настоящей работе в качестве основного метода выбрана алгебра процессов “PEPA” (Performance Evaluation Process Algebra) [11], позволяющая выполнять моделирование параллельных систем, имеющих определенные точки синхронизации, композицию моделей и использующая формальный метод описания модели.

Алгебра процессов “PEPA” — это удобное средство моделирования взаимодействующих процессов, предоставляющее структурное описание процессов, что потенциально можно использовать для введения в

модель дополнительных данных, расширяющих область ее применения. Краткий обзор алгебры "РЕРА" [11], а также ее весовое расширение приведены в работе [12].

Модели пропускной способности подсистемы памяти. В современных СУБД применяются различные методы снижения числа обращений к дисковым накопителям, в частности, за счет применения различных кэш-методов. Кроме того, как правило, архитектуры дисковых подсистем ВС достаточно однообразные. Поэтому при переходе от одной ВС к другой необходимо учитывать различия в особенностях организации подсистемы памяти.

В настоящее время в семействе ВС с архитектурой x86-x64 принципиально следует различать методы организации доступа к памяти по схеме AMD Athlon 64, AMD Opteron, Intel Xeon/Pentium 4.

Рассмотрим модели пропускной способности шин памяти и оценим коэффициент влияния метода обращения к памяти на параллельно выполняемые программы.

Принятые обозначения: ПУ — процессорный узел (как одиночный ЦП, так и современные многоядерные ЦП, в одном корпусе которых фактически совмещены несколько ЦП с использованием единого интерфейса доступа); КП — контроллер памяти, реализованный как отдельный функциональный модуль; n — количество каналов доступа к памяти.

Рассмотрим некоторые типовые схемы (рис. 1, 2).

Построим алгебраические модели со следующими параметрами: k_1 — пропускная способность внутренней шины (FSB или HyperTransport); k_2 — суммарная пропускная способность шины памяти. Примем также допущение о том, что в момент обращения процессора к памяти происходит последовательное блокирование только одной шины в один момент времени.

На основе сказанного построены следующие модели.

Модель архитектуры Intel (Xeon, Pentium 4):

$$Proc = (\mathsf{cpubus},\ \mathsf{k}_2).(\ \mathsf{membus},\ \mathsf{k}_1\ ).Proc;\ Sys = \prod^{N} Proc,$$

где действие cpubus определяет обращение по шине процессора, действие membus — обращение к ОЗУ; N — число занятых процессоров в многопроцессорной конфигурации. Дополнительные параметрические ограничения модели:

$$SYS_{RS} = [1]\,, \quad SYS_{ex} = \left[\begin{array}{ccc} 0 & 0 & 1 \end{array}\right],$$

$$act_{cpubus} = \left[\begin{array}{cc} 1 & 0 \end{array}\right], act_{membus} = \left[\begin{array}{cc} 1 & 0 \end{array}\right].$$

Модель архитектуры AMD Athlon 64 необходимо рассматривать отдельно, поскольку классическая многопроцессорная конфигурация

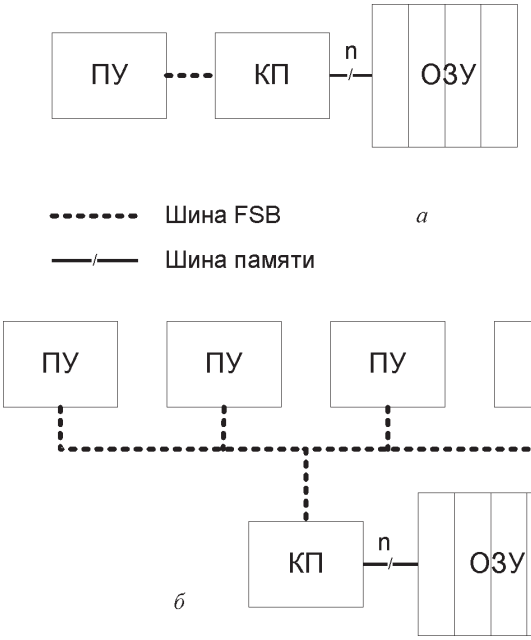

Рис. 1. Типовая схема организации доступа к ОЗУ Intel Pentium 4 (*а*) и Intel Xeon (*б*)

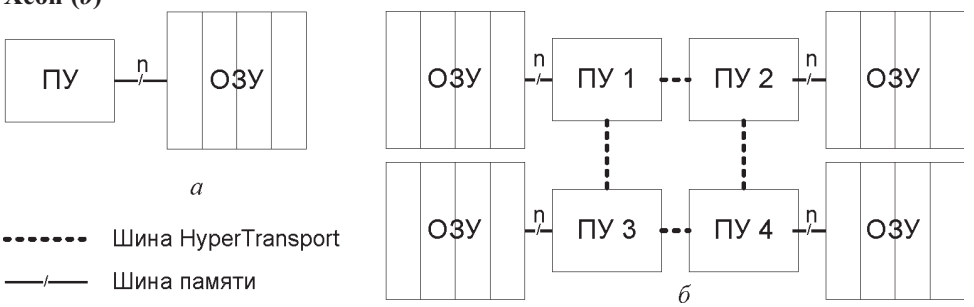

Рис. 2. Типовая схема доступа к ОЗУ AMD Athlon 64 (*а*) и AMD Opteron (*б*)

с отдельными процессорными узлами не предусмотрена, однако, возможность использования многоядерного процессорного узла требует оценки снижения общей производительности при параллельной работе нескольких ядер. Данная архитектура отличается от архитектуры Intel отсутствием шины процессора. Соответственно, модель принимает следующие параметры: $k_2 \gg k_1$, $SYS_{RS} = [1]$, $SYS_{ex} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, $act_{cpubus} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, $act_{membus} = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

Модель архитектуры AMD Opteron x4. К сожалению, в связи с отсутствием регулярности построения многопроцессорных ВС данной архитектуры, создание обобщенной модели затруднено. Рассмотрим модель, содержащую 4 процессорных узла, соединенных шинами HyperTransport по кругу (см. рис. 2).

Модель состоит из процессов, описывающих обращения каждого процессора ко всем возможным ОЗУ. Индекс процесса соответствует

номеру процессорного узла на схеме.

$\mathsf{Proc}_1 = (\mathsf{case},\ \mathsf{v}_{11}).\mathsf{Proc}_{11} + (\mathsf{case},\ \mathsf{v}_{12}).\mathsf{Proc}_{12} + (\mathsf{case},\ \mathsf{v}_{13}).\mathsf{Proc}_{13} + (\mathsf{case},\ \mathsf{v}_{14}).\mathsf{Proc}_{14};$

$\mathsf{Proc}_{11} = (\mathsf{membus1},\ \mathsf{k1}).\mathsf{Proc}_1;$

$\mathsf{Proc}_{12} = (\mathsf{cpubus}_{12},\ \mathsf{k2}).(\ \mathsf{membus}_2,\ \mathsf{k1}).\mathsf{Proc}_1;$

$\mathsf{Proc}_{13} = (\mathsf{cpubus}_{13},\ \mathsf{k2}).(\ \mathsf{membus}_3,\ \mathsf{k1}).\mathsf{Proc}_1;$

$\mathsf{Proc}_{14} = (\mathsf{cpubus}_{12},\ \mathsf{k2}).(\ \mathsf{cpubus}_{24},\ \mathsf{k2}).(\ \mathsf{membus4},\ \mathsf{k1}).\mathsf{Proc}_1 + (\mathsf{cpubus}_{13},\ \mathsf{k2}).(\ \mathsf{cpubus}_{34},\ \mathsf{k2}).(\ \mathsf{membus4},\ \mathsf{k1}).\mathsf{Proc}_1;\ \ldots\ \mathsf{Proc}_4 =$

$= (\mathsf{case},\ \mathsf{v}_{41}).\mathsf{Proc}_{41} + (\mathsf{case},\ \mathsf{v}_{42}).\mathsf{Proc}_{42} + (\mathsf{case},\ \mathsf{v}_{43}).\mathsf{Proc}_{43} + (\mathsf{case},\ \mathsf{v}_{44}).\mathsf{Proc}_{44};$

$\mathsf{Proc}_{44} = (\mathsf{membus}_4,\ \mathsf{k}_1).\mathsf{Proc}_4;$

$\mathsf{Proc}_{42} = (\mathsf{cpubus}_{24},\ \mathsf{k}_2).(\mathsf{membus}_2,\ \mathsf{k}_1).\mathsf{Proc}_4;$

$\mathsf{Proc}_{43} = (\mathsf{cpubus}_{34},\ \mathsf{k}_2).(\mathsf{membus}_3,\ \mathsf{k}_1).\mathsf{Proc}_4;$

$\mathsf{Proc}_{41} = (\mathsf{cpubus}_{24},\ \mathsf{k}_2).(\mathsf{cpubus}_{12},\ \mathsf{k}_2).(\mathsf{membus}_1,\ \mathsf{k}_1).\mathsf{Proc}_4 + (\mathsf{cpubus}_{34},\ \mathsf{k}_2).(\mathsf{cpubus}_{13},\ \mathsf{k}_2).(\mathsf{membus}_1,\ \mathsf{k}_1).\mathsf{Proc}_4;$

$Sys = \{Proc_1||Proc_i||\ldots||Proc_N\}$.

Коэффициенты v_{ij} определяют вероятности обращения к модулям ОЗУ, подключенным к разным процессорным узлам. При оценке производительности ИС, не использующих оптимизацию размещения данных для архитектуры ccNUMA, данные коэффициенты могут быть одинаковы, причем $v_{ij} \gg k_2, k_1$. Действия membus_{1-4} определяют обращения к модулям ОЗУ соответствующего ПУ, действия cpubus_{12}, cpubus_{13}, cpubus_{24}, cpubus_{34} определяют обращения по шине HyperTransport между соответствующими двойным индексам ПУ (направление передачи учитывать нецелесообразно) (табл. 1). Весовые коэффициенты следующие:

$$SYS_{RS} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad SYS_{ex} = \begin{bmatrix} 0 & 0 & 1 \\ & \ldots & \\ 0 & 0 & 1 \end{bmatrix}.$$

Таблица 1

Значения потребления ресурсов по каждому действию

Наименование действия	Значение веса ресурса соответствующей координаты. Нормирующие коэффициенты – 0							
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***
membus_1	1	0	0	0	0	0	0	0
membus_2	0	1	0	0	0	0	0	0
membus_3	0	0	1	0	0	0	0	0
membus_4	0	0	0	1	0	0	0	0
cpubus_{12}	0	0	0	0	1	0	0	0
cpubus_{13}	0	0	0	0	0	1	0	0
cpubus_{24}	0	0	0	0	0	0	1	0
cpubus_{34}	0	0	0	0	0	0	0	1

Методика преобразования моделей для различных ВС. Для использования особенностей приведенных архитектур подсистем памяти необходимо снизить размерность моделей. Поскольку весовое расширение предусматривает параметрическое ограничение моделей, целесообразно использовать именно этот вариант иерархического построения моделей, вводя результаты модели более низкого уровня через весовые коэффициенты.

Установим следующий порядок определения коэффициентов функции коррекции интенсивностей марковской цепи.

1. Для моделей памяти проводим моделирование случаев всех комбинаций занятых процессоров. Анализируемый параметр — производительность по действиям *membus* или *case* в случае модели Opteron.

2. Разные результаты по равному числу процессоров усредняем.

3. Анализируем полученные значения производительности, число которых равно числу установленных процессоров, предварительно приведя эти значения к удельной производительности в расчете на один процессор. Обозначим полученные значения как Thr_n, где n — число одновременно активных процессоров. Определим коэффициент деградации $k_{\text{ОЗУ}}(n)$ по формуле $k_{\text{ОЗУ}} = \dfrac{Thr_n}{Thr_1}$.

Полученный коэффициент $k_{\text{ОЗУ}}$ применяется для расчета коэффициента $k^3_{i'}$ в действиях моделей более высокого уровня, которые описывают потенциально одновременное обращение к подсистеме памяти.

Оценку значения параметров функции коррекции интенсивностей [12]

$$\lambda' = \lambda \left[\exp\left(- \left(k^2_{i,j} \left(\frac{\sum\limits_w rs^w_{i,j}}{RS_{i,j}} - 1 \right) \right)^{k^1_{i,j}} \right) \left(\frac{RS_{i,j}}{\sum\limits_w rs^w_{i,j}} \right)^{k^3_{i,j}} \right]$$

необходимо проводить из следующего предположения: $t_{act}(n) = t_{CPU} + t_{MEM}(n) + t_{\text{н.р.}}$, где t_{act} — время выполнения некоторого действия; t_{CPU} — время работы процессора; t_{MEM} — время, затраченное на обращение к ОЗУ; $t_{\text{н.р}}$ — время, затраченное на иные, накладные расходы, не детализируемые в рамках модели. Как было рассмотрено ранее, время обращения к памяти — это функция числа параллельных обращений.

Для упрощения расчетов аппроксимируем функцию деградации интенсивности $f_{\text{дегр}}(n) = \dfrac{\lambda_{act}(n)}{\lambda_{act}(1)} = \dfrac{1/\left(t_{CPU} + t_{MEM}(n) + t_{\text{н.р}}\right)}{1/\left(t_{CPU} + t_{MEM}(1) + t_{\text{н.р}}\right)}$ степенной функцией вида $\left(\dfrac{1}{n}\right)^{k3}$. Преобразуем функцию деградации в форму

$$f_{\text{дегр}}(n) = \left[\frac{t_{CPU} + t_{\text{н.р}}}{t_{CPU} + t_{MEM} + t_{\text{н.р}}} + \frac{t_{MEM}/k_{\text{ОЗУ}}(n)}{t_{CPU} + t_{MEM} + t_{\text{н.р}}}\right]^{-1} = \\ = \left(\%_{CPU} + \frac{\%_{MEM}}{k_{\text{ОЗУ}}(n)}\right)^{-1},$$

пренебрегая $t_{\text{н.р}}$. Таким образом,

$$k_i^3 = \frac{\log\left(\%_{CPU} + \dfrac{\%_{MEM}}{k_{\text{ОЗУ}}(n)}\right)}{\log(n)}$$

или

$$k_i^3 = \frac{\log\left[1 + \%_{MEM}\left(\dfrac{1}{k_{\text{ОЗУ}}(n)} - 1\right)\right]}{\log(n)}.$$

Алгебраические модели. *Модель выполнения запросов.* Рассмотрим ИС как замкнутую систему, в которой существуют конечное множество запросов и конечное множество клиентов. Такие допущения возможны с учетом того, что при конкретном приложении СУБД, как правило, не возникают произвольные по структуре запросы пользователей и поток запросов пользователей в целом известен.

Функционирование СУБД описывается двумя группами процессов — внутренних, представляющих собой поведение обработчиков элементарных операций, описывающих особенности реализации СУБД, взаимные блокировки процессов при выполнении этих элементарных операций, а также внешних, представляющих собой описания последовательностей выполнения элементарных операций, полученных непосредственно из физического плана выполнения, или полные процессы взаимодействия клиента и сервера как обработчика физических планов.

Модель системы клиент-сервер (под клиентом подразумеается источник запросов, под сервером — обработчик запросов) должна включать в себя следующие фазы: передачу данных по каналу связи, выполнение запроса, возврат данных по каналу связи, обработку ответа (рис. 3, *а*). В ряде случаев для упрощения модели время обработки ответа клиентом можно суммировать с временем приема/передачи данных. При исследовании предельной производительности сервера СУБД время обработки отклика клиентом и время приема/передачи можно принять равными нулю.

При транзакционной обработке предполагаетсч, что данные, блокированные в результате запроса, могут быть не разблокированы в рамках этого же запроса. Завершение обработки происходит только по специальной команде или в рамках модели после определенного

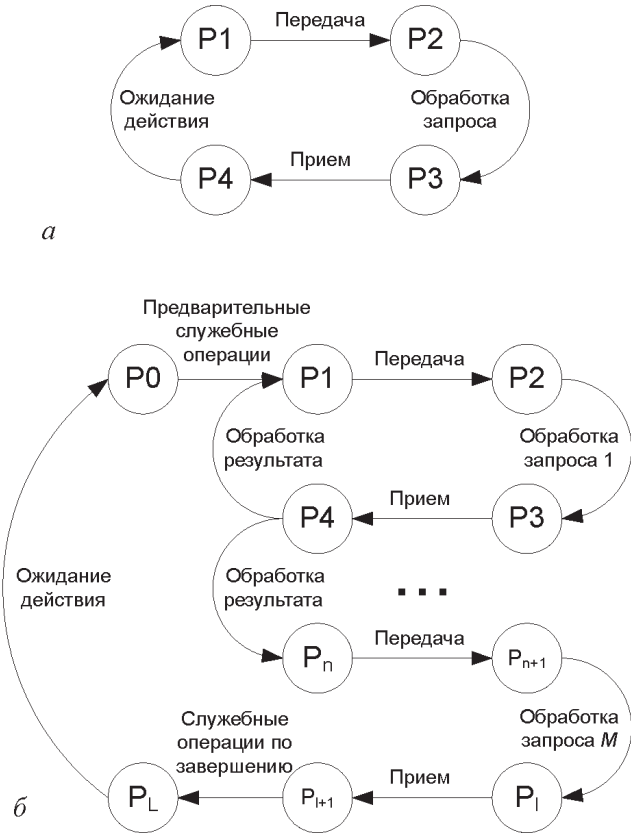

Рис. 3. Графы переходов при обработке элементарных запросов:
а — одиночные запросы; *б* — несколько запросов в рамках транзакции

числа циклов обработки, кроме того, в общем виде выполняемые запросы не одинаковы, следовательно в рамках одной транзакции может выполняться несколько различных запросов (рис. 3, *б*).

Моделирование ИС с упрощенной архитектурой. При моделировании СУБД в некоторых случаях применима модель с двумя типами взаимоисключающих операций, поскольку такая модель представляет собой наиболее общий случай выполнения запросов некоторым устройством обработки. Предполагается, что выполнение запросов различается по принципу эксклюзивности операций, причем при выполнении запросов чтения их взаимное влияние не рассматривается. На рис. 4 приведен обобщенный алгоритм обслуживания запросов сервером СУБД. При анализе заявки от клиента определяется тип операции (чтение — неэксклюзивная заявка, запись — эксклюзивная). Если поступившая заявка требует операцию записи, то сервер ожидает завершение всех текущих операций и выполняет данную заявку. Если же поступила заявка на операцию чтения (если сервер не выполняет в данный момент операцию записи) заявка поступает на выполнение. Иначе производится окончание текущей операции записи.

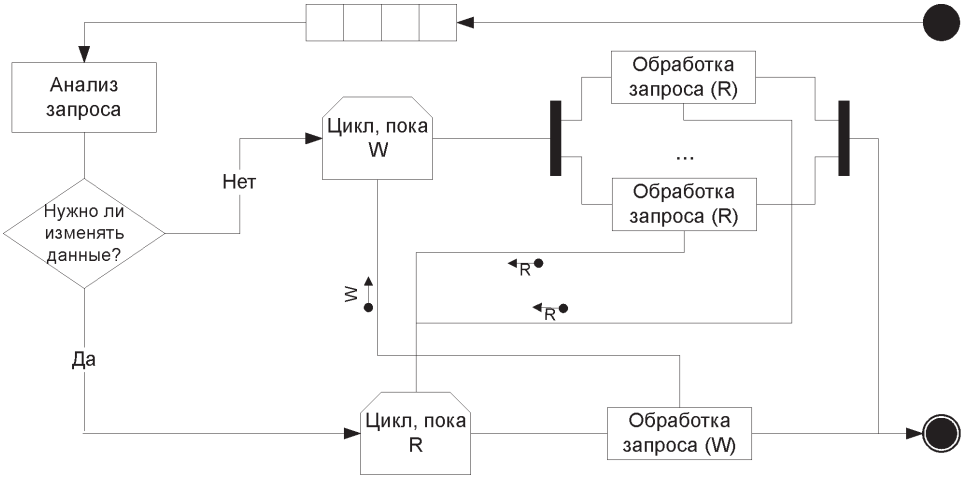

Рис. 4. Алгоритм обслужмвания запросов

Составим модель в терминах алгебры процессов PEPA для сервера СУБД и контура клиента. В соответствии с ранее описанным принципом построения моделей выделим процессы *SrvR* и *SrvW*, описывающие обработку операций сервером с интенсивностями r и w, соответственно. Клиенты описываются процессом *Client*, генерирующим заявки на чтение (inR) и запись (inW) с интенсивностями *n*in* и $(1-n)$**in* соответственно. Кроме того, дополнительную задержку вносит отображение клиентом полученной от сервера информации, моделируемое действием display с интенсивностью *d*. Причем при моделировании предельной нагрузки сервера данное действие следует исключить. Составленная модель имеет вид

SrvR =(beginR, T).(lockR, *l*).(actR, *r*).(unlockR, *l*).(endR, T).*SrvR*;
SrvW =(beginW, T).(lockW, *l*).(actW, *w*).(unlockW, *l*).(endW, T).*SrvW*;

Sem_0 = (lockR, T).Sem_1 + (lockW, T).*SemW*;
Sem_i = (lockR, T).Sem_{i+1}+ (unlockR, T).Sem_{i-1}; при $i<N$
Sem_N = (unlockR, T).Sem_{N-1};
SemW = (unlockW, T).Sem_0;

ClientProc = (display, d).Client;

Client =
(inR, n_1**in*).(beginR, T).(endR, T).*ClientProc* +
(inW, n_2**in*).(beginW, T).(endW, T).*ClientProc*;

$$Sys = \left(\prod_{i=1}^{M} Client\right) \underset{L_1}{\bowtie} \left(\prod^{N} SrvR||SrvW\right) \underset{L_2}{\bowtie} Sem_0$$

$L_1 = \{beginR, beginW, endR, endW\}$,
$L_2 = \{lockR, lockW, unlockW, unlockW\}$.

Детализированная модель ИС. В общем случае для оценки характеристик СУБД требуется описать модель, содержащую не только эксклюзивные/неэксклюзивные операции сервера СУБД, но и описание особенностей обработки конкретных операций, а также модели запросов клиентов ИС. Опишем модель ИПС, состоящую из модели клиента ИПС и модели сервера СУБД.

Клиент ИПС. Модель клиента описывает выполняемые клиентом действия в виде последовательностей высокоуровневых операций. При этом не рассматриваются ни особенности СУБД, ни состав данных.

$$\begin{aligned}\text{Client} = {} & (\text{in}, \lambda_{\text{search}}).\text{Search} + \\ & (\text{in}, \lambda_{\text{rubric}}).\text{GetRubric} + \\ & (\text{in}, \lambda_{\text{read}}).\text{Read} + \\ & (\text{in}, \lambda_{\text{add}}).\text{Add} + \\ & (\text{in}, \lambda_{\text{change}}).\text{Change} + \\ & (\text{in}, \lambda_{\text{del}}).\text{Del};\end{aligned}$$

$$\begin{aligned}\text{Search} = {} & (\text{beginSearchText},T).(\text{endSearchText},T).(\text{net}, \lambda_{\text{net}}). \\ & (\text{beginClear},T).(\text{endClear}, T).(\text{net}, \lambda_{\text{net}}).\text{Client};\end{aligned}$$

$$\begin{aligned}\text{GetRubric} = {} & (\text{beginSearch},T).(\text{endSearch},T).(\text{net}, \lambda_{\text{net}}). \\ & (\text{beginClear},T).(\text{endClear}, T) .(\text{net}, \lambda_{\text{net}}).\text{Client};\end{aligned}$$

$$\begin{aligned}\text{Read} = {} & (\text{beginLock}, T).(\text{endLock},T). \\ & (\text{beginRead}, T).(\text{beginRead}, T).(\text{net}, \lambda_{\text{net}}).\text{Client};\end{aligned}$$

$$\begin{aligned}\text{Add} = {} & (\text{begin AddTrQueue}, T).(\text{end AddTrQueue},T). \\ & (\text{beginAdd}, T).(\text{endAdd},T).(\text{net}, \lambda_{\text{net}}).\text{Client};\end{aligned}$$

$$\begin{aligned}\text{Change} = {} & (\text{beginLock}, T).(\text{endLock},T). \\ & (\text{begin AddTrQueue}, T).(\text{end AddTrQueue},T). \\ & (\text{beginLock}, T).(\text{endLock},T). \\ & (\text{beginChange}, T).(\text{beginChange}, T).(\text{net}, \lambda_{\text{net}}).\text{Client};\end{aligned}$$

$$\begin{aligned}\text{Del} = {} & (\text{begin AddTrQueue}, T).(\text{end AddTrQueue},T). \\ & (\text{beginDel}, T).(\text{endDel},T). (\text{net}, \lambda_{\text{net}}).\text{Client}.\end{aligned}$$

Действие net с параметром λ_{net} описывает процесс передачи данных по сетевому интерфейсу. При анализе предельной производительности допустимо принять $\lambda_{\text{net}} = 0$, т.е. исключить из модели.

Сервер СУБД. Модель сервера СУБД описывает внутреннюю структуру СУБД, при этом посредством среднего времени выполнения операций учитывает состав данных. Кроме того, эта модель учитывает внутреннее параллельное расщепление процессов обработки.

λ_{inst} »1;

// Семафор операций:
Sem = (lockR, T).SemR1 + (lockW, T).SemW$_1$;

SemR$_1$ = (lockR, T).SemR$_2$ + (unlockR, T).Sem;
SemR$_i$ = (lockR, T).SemR$_{i+1}$ + (unlockR, T).SemR$_{i-1}$; // 1<i<N$_{\text{par}}$
SemR$_{Npar}$ = (unlockR, T).SemR$_{\text{Npar}-1}$;

SemW$_1$ = (unlockW, T).Sem;

// описание внутренних операций

SearchText = (beginSearchText, λ_{inst}).(lockR, λ_{inst}).
(search, T).
(searchText, T).
(prepareResult, $\lambda_{\text{pr - text}}$).
(unlockR, λ_{inst}).(endSearchText, λ_{inst}).SearchText;

Search = (beginSearch, λ_{inst}).(lockR, λ_{inst}).
(search, T).
(prepareResult, $\lambda_{\text{pr - field}}$).
(unlockR, λ_{inst}).(endSearch, λ_{inst}).Search;

Clear = (beginClear, λ_{inst}).(lockR, λ_{inst}).
(clear, λ_{clear}).
(unlockR, λ_{inst}).(endClear, λ_{inst}).Clear;

Read = (beginRead, λ_{inst}).(lockR, λ_{inst}).
(read, λ_{read}).
(unlockR, λ_{inst}).(beginRead, λ_{inst}).Read;

Add = (beginAdd, λ_{inst}).(lockW, λ_{inst}).
(add, λ_{add}).
(unlockW, λ_{inst}).(endAdd, λ_{inst}).Add;

Change = (beginChange, λ_{inst}).(lockW, λ_{inst}).
(change, λ_{change}).
(unlockW, λ_{inst}).(beginChange, λ_{inst}).Change;

Del = (beginDel, λ_{inst}).(lockW, λ_{inst}).
(del, λ_{del}).
(unlockW, λ_{inst}).(endDel, λ_{inst}).Del;

AddTrQueue =(beginAddTrQueue, λ_{inst}).(lockR, λ_{inst}).
(addQueue, λ_{queue}).
(unlockR, λ_{inst}).(endAddTrQueue, λ_{inst}).Commit;

Hnd_SearchText = (searchText, $\lambda_{\text{search - text}}$).Hnd_SearchText;
Hnd_Search = (search, λ_{search}).Hnd_Search;

$$SearchDef = (\overbrace{Search \underset{L_1}{\triangleright\!\triangleleft} Search \ldots Search}^{N_1}),$$ где

$$L_1 = \{beginSearch, lockR, prepareResult, endSearch, unlockR\},$$

$$SearchTextDef = (\overbrace{SearchText \underset{L_2}{\triangleright\!\triangleleft} SearchText \ldots Search Text}^{N_1}),$$ где

$$L_2 = \{beginSearch, lockR, prepareResult, endSearch, unlockR\}.$$

Уравнение системы

$$Sys = (\overbrace{Client||...||Client}^{N_{client}}) \underset{L_3}{\triangleright\!\triangleleft} ($$
$$(\overbrace{SearchTextDef|| \ldots ||SearchTextDef}^{N_{par}}) \Rightarrow$$
$$\Rightarrow ||(\overbrace{SearchText|| \ldots ||SearchText}^{N_{par}})||$$
$$(\overbrace{Clear|| \ldots ||Clear}^{N_{par}})||(\overbrace{AddTrQueue|| \ldots ||AddTrQueue}^{N_{par}})||$$
$$(\overbrace{Add|| \ldots ||Add}^{N_{par}})||(\overbrace{Change|| \ldots ||Change}^{N_{par}})||(\overbrace{Del|| \ldots ||Del}^{N_{par}})||$$
$$(\overbrace{Hnd\_SearchText|| \ldots ||HndSearchText}^{N_{par2}}) \Rightarrow$$
$$\Rightarrow ||(\overbrace{Hnd\_Search|| \ldots ||HndSearch}^{N_{par2}})) \underset{L_4}{\triangleright\!\triangleleft} Sem)$$

$$L_3 = \left\{ \begin{array}{l} beginSearch, beginSearchText, beginLock, beginAdd, \\ beginChange, beginDel, beginAddTrQueue, \\ \\ endSearch, endSearchText, endLock, endAdd, endChange, \\ endDel, endAddTrQueue. \end{array} \right\}$$

Полученная модель описывает верхний уровень представления, включающий в себя внутренние алгоритмы обработки на концептуальном уровне. Возможна дальнейшая детализация модели. Модель позволяет оценить как предельные характеристики производительности ИС по каждому типу запросов, так и их влияние на общую производительность. Главный недостаток такой модели — сложность решения марковской цепи, полученной на ее основе, поскольку ее размерность будет достаточно большой.

Модели подсистем сервера СУБД. Для устранения проблем размерности в ряде случаев применимо моделирование отдельных под-

систем сервера СУБД. Рассмотрим моделирование подсистемы полнотекстового поиска в ОСУБД ODB-Jupiter для оценки производительности параллельной и последовательной схем обслуживания запроса поиска.

Подсистема поиска обеспечивает выборку списков объектов, соответствующих заданному условию из индексов заданных типов объектов с учетом объектного наследования. Таким образом, поиск объектов включает в себя поиск одного элемента запроса по конкретному индексу, обработку результатов поиска для одного индекса и по всем индексам (табл. 2). Особенность ОСУБД ODB-Jupiter в том, что каждый тип объектов, соответственно и индексы атрибутов данного типа, могут быть размещены в индивидуальных хранилищах данных, что позволяет выполнить параллельную обработку в рамках разных хранилищ.

Таблица 2

Принятые обозначения

Наименование действия	Наименование соответствующей интенсивности	Комментарий
Search	*l_search*	Обработка результатов всего поиска
Search_idx_item	*l_search_idx_item*	Поиск одного элемента в рамках индекса
loop_search_idx	*inst·(1-s1), inst·s1*	Фиктивное действие для организации цикла индексов

Последовательная модель:

Search = (search, λ_{search}).Search1;

Search1 = (search_idx_item, $\lambda_{\text{search\_idx\_item}}$).Search2;

Search2=(loop_search, ll*(1-s1)).Search + (loop_search, ll*s1).Search1;

$$Sys = \left(\prod^{N_1} Search\right).$$

Параллельная модель:

Search = (search, λ_{search}).Search1;

Search1 = (search_idx_item, T).Search;

SIdx = (search_idx_item, $\lambda_{\text{search\_idx\_item}}/N_3$).SIdx;

$$Sys = \left(\prod^{N_1} Search\right) \underset{\{search\_idx\_item\}}{\triangleright\!\triangleleft} \left(\prod^{N_2} SIdx\right);$$

N_1, N_2 и N_3 — число клиентов, параллельных обработчиков и параллельных хранилищ, $s_1 = 1 - 1/N_3$.

Введем размерность 1 для весового вектора, а также для всех моделей определим следующие значения: $w^1_{\text{search\_idx\_item}} = 10$, $W^1_{sys} = 10$, $k^1_1 = 0$, $k^2_1 = 0$, $k^3_1 = 1$.

Оценку точности представления данными моделями подсистемы поиска проведем позднее.

Определение базовых параметров моделей. Важный момент в математическом моделировании — это выбор параметров моделей. При рассмотрении моделей ИС необходимо учитывать множество параметров, зависящих как от используемого для функционирования ИС оборудования ВС, так и от особенностей программной реализации ОС и ИС. Прямой расчет параметров по физическим характеристикам оборудования невозможен из-за отсутствия прямой связи многих из них по отдельности с производительностью в целом. Как показывает анализ средств измерения производительности, отсутствуют единые методы оценки интегральной характеристики производительности оборудования в отношении некоторого универсального ПО. Так, конкурирующие производители ЦП часто демонстрируют результаты тестирования серверов на базе своих процессоров, превосходящие результаты тестирования серверов на базе процессоров конкурентов, на одних задачах, в то время как конкуренты демонстрируют прямо противоположную картину на несколько отличающихся задачах. Таким образом, наиболее достоверные данные можно получить только по результатам проведения натурного эксперимента на программах, наиболее близко соответствующих коду моделируемой ИС, а в идеале — по результатам выполнения на какой-либо известной ВС реальной ИС. Следует заметить, что степень необходимой детализации измерения ИС при этом зависит от детализации модели.

Измерение времени выполнения функций сервера СУБД следует проводить в условиях, наиболее приближенных к ИС, в рамках которой СУБД будет эксплуатироваться. Кроме того, следует учитывать, что в ИС используется конечный набор запросов к СУБД, обусловленный логикой ее работы, в то время как конечные пользователи ИС обычно могут лишь параметризовать некоторые из этих запросов.

Предложим следующую схему получения необходимых данных.

1. Обеспечить непрерывный поток заявок в рамках ИС, причем для создания потока наиболее адекватных запросов целесообразно использовать клиентов ИС.

2. Для получения точных данных о характеристиках выполнения, необходимо внедрить в сервер СУБД соответствующие средства измерения и экспорта данных. На рис. 5 приведена упрощенная схема измерения, используемая при анализе ОСУБД ODB-Jupiter.

Для управления процессом нагрузочного тестирования применялся специальный модуль, обеспечивающий управление клиентами ИС и формирующий заданные команды ИС, которые (в соответствии с

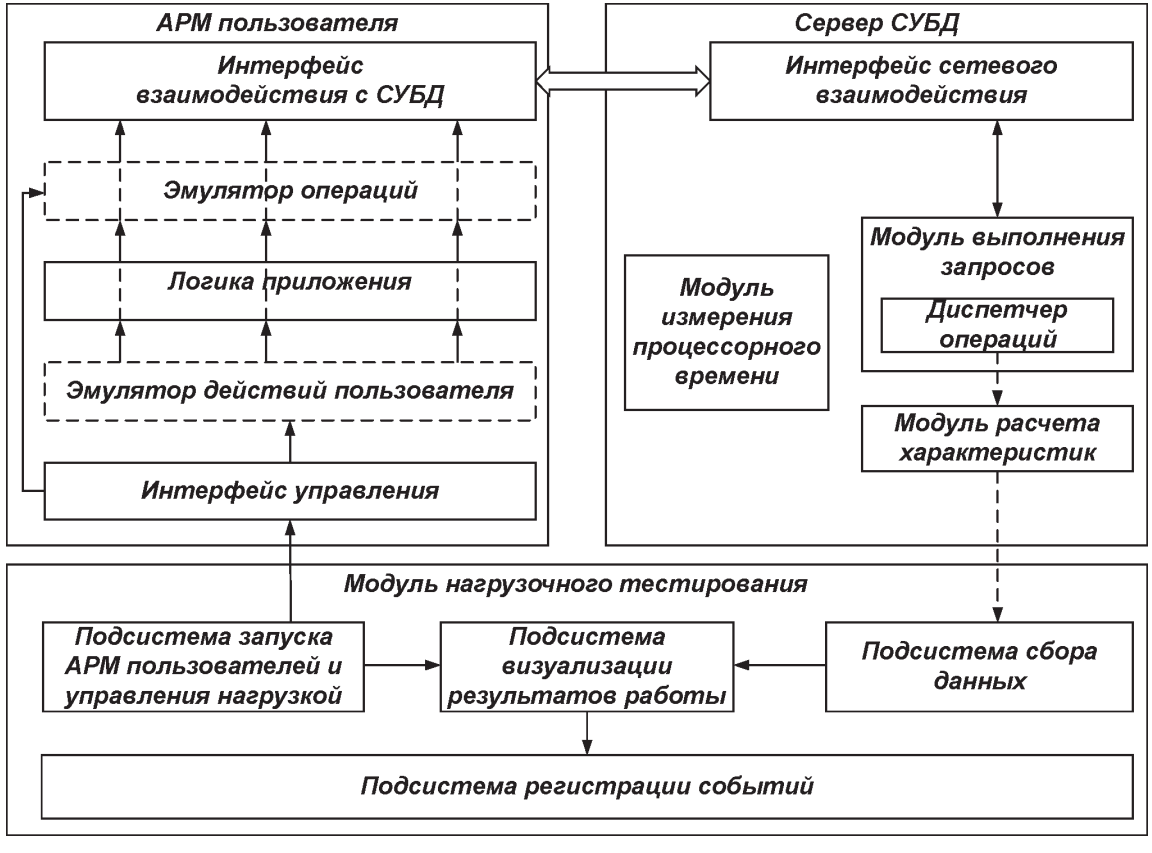

Рис. 5. Схема комплекса нагрузочного тестирования

логикой ИС) представляют собой набор конкретных физических команд сервера СУБД. В сервер СУБД встроены средства измерения малых временных интервалов для измерения времени выполнения элементарных операций, соответствующим образом модифицирован исходный код сервера СУБД для обеспечения измерения времени их выполнения, а также встроены средства для передачи характеристик выполнения запросов внешним процессам, типа модуля нагрузочного тестирования для создания замкнутой системы управления процессом тестирования. Кроме того, функцией модуля нагрузочного тестирования также является сбор данных о состоянии ВС и ОС. Итоги такого тестирования на конкретных операциях ИС, генерируемых клиентами ИС, следующие:

— средние и предельные значения времени выполнения внутренних операций сервера ИС для использования в моделях в качестве среднего времени выполнения действий (т.е. обратных величин — интенсивностей переходов);

— средние и предельные характеристики производительности по каждому типу запросов;

— характеристики потребления системных ресурсов ВС и ОС для выполнения параметризации моделей по результатам моделирования.

Оценка точности моделирования подсистемы полнотекстового поиска. Исследуем результаты вычислительного эксперимента моделей и ИС, проверим точность описания математических моделей, параметризованных на основе экспериментальных данных ИС, работающей в соответствующем режиме на соответствующем оборудовании, а также точность моделей, полученных путем предсказания свойств будущей ИС, параметризованных по результатам экспериментальных данных другой ВС.

Цель эксперимента — анализ эффективности параллельного обслуживания запросов полнотекстового поиска документов ИПС “Обзор СМИ”. Для исключения избыточного влияния факторов, снижающих степень параллельной обработки, были установлены следующие условия проведения эксперимента.

Тестовая БД состоит из восьми хранилищ данных, что обеспечивает параллельный поиск данных (до восьми каналов одновременно). Каждое хранилище содержит около 30 тыс. документов со средним объемом текстовых данных 2 кБ.

Поиск проводится на малом наборе текстовых запросов, требующих выполнения большого объема операций позиционирования по индексам данных. Размер кэшей индексных данных подобран так, чтобы все необходимые данные полностью вмещались в отведенное

пространство памяти, а обращения к дисковым накопителям были минимизированы — обеспечивается анализ эффективности параллельной обработки запросов в режиме процессор–память.

Запросы формируются модифицированными программами-клиентами ИПС “Обзор СМИ” так, чтобы каждый клиент циклически обеспечивал непрерывный поток запросов на сервер с синхронизацией по окончанию обработки каждого запроса. Число одновременно обслуживаемых клиентов в результатах измерения соответствует числу одновременно работающих клиентов комплекса, поскольку время работы самого клиента намного меньше времени обработки запроса.

Данные о производительности получены непосредственно от сервера СУБД в результате обработки реальных запросов. Время работы для всех ВС для исключения влияния переходных процессов было одинаковым и принято равным 600 с.

Алгоритм обработки полнотекстовых запросов включает в себя три фазы — предварительную, фазу поиска и фазу подготовки результатов. Внутренняя параллельная обработка предусмотрена только на этапе фазы поиска, поэтому для анализа будем рассматривать только среднее время поиска в одном хранилище данных и время остальной обработки. Для анализа этих значений времени код программы ИС был размечен в соответствии с ранее представленной методикой измерения времени выполнения.

Модель подсистемы поиска была приведена ранее. Проведем подбор параметров. Наиболее важным для дальнейшего использования модели в целях предсказания характеристик будущей параллельной системы является корректное разделение составляющих времени, затраченного на работу процессора и на обмен данными с ОЗУ.

Ранее предлагалась коррекция интенсивностей марковской цепи степенной функцией с коэффициентом

$$k_i^3 = \frac{\log\left[1 + \%_{\text{МЕМ}}\left(1/k_{\text{ОЗУ}}(n) - 1\right)\right]}{\log(n)}$$

и $\%_{CPU} = 1 - \%_{\text{МЕМ}}$. В начальный момент коэффициент k_i^3 подбирается эмпирически, что позволяет оценить распределение долей времени работы ЦП и времени обмена ОЗУ. Из приведенного выражения можно определить, что $\%_{\text{МЕМ}} = \dfrac{n^{k_i^3} - 1}{1/k_{\text{ОЗУ}}(n) - 1}$.

Учитывая, что известны суммарное время работы процессоров, пропускная способность памяти и их производительность, можно пересчитать время выполнения анализируемых действий для предполагаемых систем.

Исходные данные для моделирования приняты по результатам измерения ВС на базе AMD Athlon X2 4400+, ОЗУ 2048 MB, HDD — 1 SATA 250 GB.

По результатам измерения реальной системы получены значения времени выполнения функций (табл. 3).

В соответствии с ранее описанной методикой преобразования моделей для различных архитектур получаем значения интенсивностей и коэффициента функции коррекции (табл. 4).

Таблица 3

Средние значения времени выполнения операций при полнотекстовом поиске

Наименование функции	Назначение	Время, мс
Search	Предварительная обработка запроса и окончательная обработка выборки данных	855
TextSearch	Выполнение специальных функций над результатами выборки индекса и диспетчеризация поиска	103
InternalSearch	Извлечение индексов, соответствующих запросу	94

Таблица 4

Значения параметров моделей

ВС	λ_{search}	$\lambda_{\text{search\_idx\_item}}$	%CPU	%MEM	k_i^3
Процессор AMD Athlon 64 X2 4400+ ОЗУ 2 ГБ, размер базы — 4 ГБ	9,67	10,64	83,5	16,5	0,22
* Процессор 4 x Opteron 880 2,4 ГГц ОЗУ 16 ГБ, размер базы — 4 ГБ	8,9	9,9	71	29	0,11
* Процессор 4 x Intel Xeon 2,6 ГГц ОЗУ 1024 МБ, размер базы — 4 ГБ	5,37	5,91	82	18	0,32

"*" ВС параметризованы по результатам реального измерения применительно к первой ВС с использованием методики преобразования моделей.

Результаты моделирования для последовательного и параллельного случаев приведены на рис. 6 и 7. В каждой группе графиков показаны значения, полученные по результатам вычислительного и реального экспериментов. Значения вычислительного эксперимента помечены сокращением "мод".

Значения ошибки моделирования приведены в табл. 5.

Сравнивая результаты моделирования параллельной схемы обработки с реальными результатами, необходимо отметить наличие систематической ошибки, растущей с увеличением числа процессоров. Это связано с тем, что в модели не учтены затраты на межпроцессорные переключения контекста выполнения операций, реализация

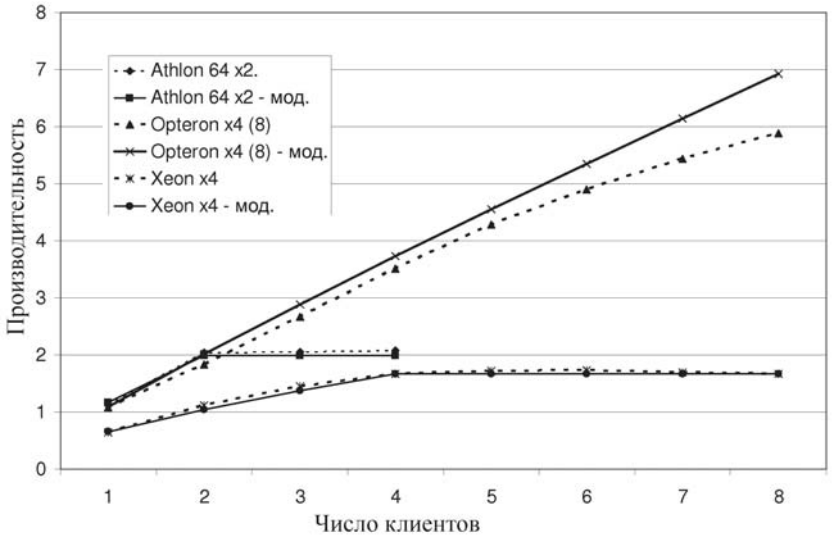

Рис. 6. Результаты последовательной схемы выполнения

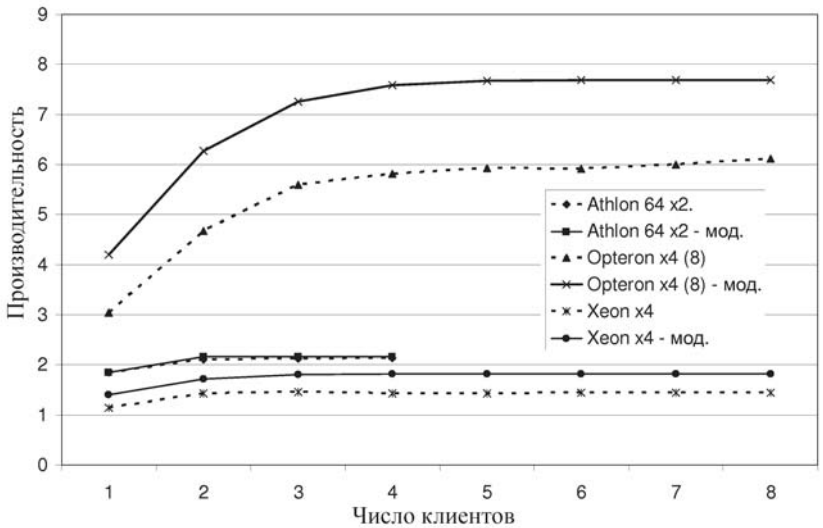

Рис. 7. Результаты параллельной схемы выполнения

алгоритмов управления доступом к страницам памяти и другие системные накладные расходы. С учетом изложенного можно сделать вывод о соответствии форм кривых производительности, что позволяет использовать модели для общей оценки производительности и влияния на нее особенностей внутренней обработки даже без введения поправочного коэффициента производительности.

Выводы. Рассмотренные метод и модели обеспечивают оценку характеристик СУБД и ИС, выполненных на их основе. Наличие

Таблица 5

Значения параметров моделей

ВС	Ошибка последовательной модели, %	Ошибка параллельной модели, %	Систематическая ошибка параллельной модели, %
Процессор AMD Athlon 64 X2 4400+	– 4–4,5	0,4–2,8	0
* Процессор 4 x Opteron 880 2,4 ГГц	1,1–17,5	22–37	29,3
* Процессор 4 x Intel Xeon 2,6 ГГц	– 7,2–0,6	20–27	25,1

* — см. сноски в табл. 4.

средств моделирования иерархических параллельных процессов позволяет использовать метод, разработанный на основе алгебры процессов “РЕРА”, в широком классе задач и, в частности, моделировать параллельные СУБД с внутренним расщеплением параллельных процессов. Полученные результаты свидетельствуют о принципиальной возможности использования данных методов для качественной и количественной оценки производительности ИС. Проблемы, связанные с решением полных моделей ИС, и, соответственно, большой размерностью промежуточных марковских цепей, являются принципиально устранимыми, но для этого требуется разработка соответствующих методов понижения размерности и фрагментарного решения. Тем не менее, приведенный метод моделирования отдельных подсистем позволяет существенно снизить размерность модели при анализе производительности ИС на конкретных операциях.

СПИСОК ЛИТЕРАТУРЫ

1. Dandamudi S. and Au Siu-Lun. Locking granularity in multiprocessor database systems / CH2968-6/0000/0268$01.00 : IEEE–1991.
2. Соколинский Л. Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. – 2001. – № 6. – С. 13–29.
3. Hillston J. A compositional approach to performance modelling // Cambridge University Press, 1996, 168 p.
4. Соколинский Л. Б. Параллельные машины баз данных // Природа. ЕНЖ РАН. – 2001. – № 8. – С. 10–17.
5. Delis A., Roussopoulos N. Performance and scalability of client-server database architectures // Proceedings of 18th International Conference on VLDB. – Vancouver, 1992. – P. 610–623.
6. Dempster E. W., Tomov N. T., Williams M. H., et. al. Modelling parallel oracle for performance prediction / Distributed and parallel databases, 13, 251–269, Kluwer Academic Publishers, 2003.

7. Goetz Graefe. Query evaluation techniques for large databases // ACM Computing Surveys, Vol. 25, No. 2, June 1993.
8. Clark G. Techniques for the construction and analysis of algebraic performance models: PhD thesis, The University of Edinburgh, 2000.
9. Bhide A. An analysis of three transaction processing architectures // Fourteenth International Conference on Very Large Data Bases (VLDB'88), August 29 – September 1, 1988, Los Angeles, California, USA, Proceedings. Morgan Kaufmann. – 1988. – P. 339–350.
10. Dumas Sophie, Gardarin Georges. A workbench for predicting the performances of distributed object architectures / Proceedings of the 1998 Winter Simulation Conference, 1998.
11. Gardarin G., Sha F., and TangZ.-H. Calibrating the query optimizer cost model of IRO-DB, an object oriented federated database system // Proc. of the Conf. on Very Large Data Bases(VLDB), pages 378–389, T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, Nandlal L. Sarda (Eds.): VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3–6, 1996, Mumbai (Bombay), India. Morgan Kaufmann 1996, ISBN 1-55860-382-4.
12. Андреев А. М., Березкин Д. В., Самарев Р. С. Применение алгебраических моделей при разработке СУБД и ИС на их основе // Информационные технологии. – 2007. – № 11.

Статья поступила в редакцию 22.05.2007

В издательстве МГТУ им. Н.Э. Баумана
в 2006 г. вышла в свет книга

Козинцев В.И.

Основы импульсной лазерной локации. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. – 512 с.

Изложены физические основы импульсной лазерной локации. Приведены сведения об оптических свойствах земной атмосферы, отражающих свойствах земной и морской поверхностей и объектов локации. Описаны эффекты, возникающие при распространении лазерных пучков в атмосфере. Рассмотрены методы расчета лазерных сигналов на трассе с отражением от неровной земной и взволнованной морской поверхностей, от светоотражателей и от объектов сложной формы. Описаны помехи в системах лазерной локации. Изложены теоретические основы приема лазерных сигналов. Приведены примеры лазерных локационных систем различного назначения и описаны их основные элементы. Содержание учебного пособия соответствует курсу лекций, который читают авторы в МГТУ им. Н.Э. Баумана.

Для студентов технических вузов, обучающихся по направлению "Оптотехника", а также для научных работников и инженеров приборостроительного профиля.

По вопросам приобретения обращаться по тел. 263-60-45;
e-mail: press@bmstu.ru