

Jet

INFO

**МАТЕРИАЛ
МАТЕРИА
НОМЕРА**

**Параллельные
архитектуры
серверов
баз данных**

А ТАКЖЕ:

- КОЛОНКА РЕДАКТОРА
- НОВОСТИ INTERNET
- ЧТО ЕЩЕ ПОЧИТАТЬ
- ПРЕДЫДУЩИЕ ВЫПУСКИ JET INFO

1
1995



Уважаемый Читатель!

Бюллетень Jet Info издается компанией Jet Infosystems около трех лет. За этот срок интерес к компании, к темам, затрагиваемым в выпусках Jet Info, к публикациям его постоянных авторов, да и к самому изданию вырос настолько, что решение сделать его выход регулярным и значительно более частым стало, на наш взгляд, логичным и оправданным. Теперь Jet Info будет выходить дважды в месяц, а Вы, Читатель, сможете получать его по почте — обычной или электронной. Стать подписчиком Jet Info очень просто — заполните анкету, помещенную в конце номера, и отправьте нам ее по почте или факсу. Все наши реквизиты Вы можете найти на обложке. Напишите нам, если Вы хотите получить какие-либо из ранее выпущенных номеров Jet Info — их список Вы найдете по соседству с анкетой.

Наш редакционный портфель уже сейчас содержит солидное количество статей. В Jet Info будут публиковаться материалы по сетевым технологиям, системам управления базами данных, статьи о продуктах и стратегических разработках компании Sun Microsystems Inc., об объектно-ориентированных технологиях и информационной безопасности. Специалисты компании Jet Infosystems, занимаясь системной интеграцией, накопили большой опыт в перечисленных областях и им есть что рассказать.

В Jet Infosystems приходят не только потому, что здесь можно приобрести компьютеры компании Sun Microsystems или сетевое оборудование компании Bay Networks, но, главным образом, потому, что здесь работают специалисты, делом доказавшие свое умение комплексно решать задачи клиентов и реально владеющие современной технологией программирования и системной интеграции. Как сказал один из наших новых партнеров: "Я к вам пришел потому, что здесь работает Глеб Ладыженский, который знает все о TUXEDO".

Предыдущие выпуски Jet Info продемонстрировали умение специалистов Jet Infosystems доходчиво излагать накопленные знания. Есть основания полагать, что это качество не будет утеряно.

На страницах бюллетеня найдется место для материалов о наших партнерах и для статей, описывающих достижения наших клиентов. Будут представлены новости, сведения о мероприятиях, проводимых компанией Jet Infosystems, и другая оперативная информация. Мы планируем отвести одну-две страницы для переписки с читателями и, в частности, для ответов на Ваши вопросы. Вполне возможно, что у Вас, Читатель, есть или будут появляться вопросы, не относящиеся непосредственно к Jet Info, но так или иначе попадающие в сферу интересов компании Jet Infosystems. Мы готовы отвечать на них как на страницах Jet Info, так и в деловой переписке (желательно, электронной).

Мы надеемся, что наше сотрудничество будет продолжаться.

*В. А. Галащенко,
главный редактор Jet Info*



Ведущий рубрики - Александр Гагин

"Позвольте рекомендоваться: меня зовут Гагиным, а это моя... — он зашнуровал на мгновение, — моя ..."

И. С. Тургенев. "Ася"

ТАЛИСМАНОМ КОРПОРАЦИИ SUN СТАЛА СОБАКА

В конце мая корпорация Sun Microsystems Inc. начала многомиллионную рекламную кампанию, в центре которой — новый образ корпорации. Сигналом к началу послужили публикации в Wall Street Journal, New York Times и других изданиях. Кампания строится вокруг огромной добродушной собаки по имени Network (Сеть). Пес буквально излучает мощь и дружелюбие, работая тем самым на образ корпорации Sun как ли-

дера сетевой индустрии. Удачно найденный основной образ делает естественными рекламные ходы типа: "Посмотрите, как много информации может переносить наша Сеть" или "Скажите, что Вам нужно, и Сеть сделает это для Вас".

Пословица гласит: "Только ведущая собака в упряжке видит что-то новое". Sun пытается и сам видеть новое, и для других выглядеть по-новому.

НЕ ПУСТЯК, НО ВСЕ РАВНО ПРИЯТНО

Федеральное подразделение компании Sun Microsystems Inc. празднует самый большой контракт в своей истории. Согласно этому контракту, в десятилетний срок компания Sun Microsystems должна поставить американским военным примерно 28 тысяч рабочих станций и серверов. Этот контракт является частью программы переоснащения армии США под названием Hardware-Software 2. Вкус победы для Sun Microsystems еще слаще от того, что предыдущим поставщиком была корпорация Miltop с рабочими станциями компании Hewlett-Packard.

Поставляемые системы обеспечат военным единую аппаратно-программную платформу — Sparc и Solaris. Номенклатура поставок весьма широка — рабочие станции, серверы, дисплеи с большими экранами, мосты, маршрутизаторы, принтеры, сканеры, безопасные операционные системы, реализации сетевых протоколов и прикладные системы. Все оборудование будет поставляться в защищенном исполнении с возможностью работы в полевых условиях.

ПЕРВЫМ ДЕЛОМ - ULTRASPARC'И...

Компания SunSoft Inc. готовит к выпуску новую версию ОС Solaris — Solaris 2.5. Предполагается, что варианты для трех аппаратных платформ — Sparc, Intel и PowerPC, полученные из единого дерева исходных текстов, будут поставляться на одном CD-ROM'е. Цены пока не определены.

Впрочем, возможность поставки трех вариантов на одном CD-ROM'е пока проблематична, поскольку выпуск варианта для процессоров Sparc (точнее,

UltraSparc) запланирован на вторую половину 1995 года, а редакция для "суперклиента" с PowerPC будет полностью готова только в первом квартале 1996 года. Причина отставания редакции для PowerPC в том, что SunSoft хочет в полном объеме задействовать возможности SMP-конфигураций, а подобная конфигурация с PowerPC 604 поступила в распоряжение специалистов SunSoft лишь в начале июня.

МЫ ВЫБИРАЕМ ULTRASPARC TOSHIBA?

24 мая 1995 года в Сан-Франциско на конференции SunWorld'95 компании SPARC Technology Business (подразделение компании Sun Microsystems) и Data General Corporation объявили, что в первой половине 1996 года операционная система DG/UX корпорации Data General

будет работать на новом 64-разрядном микропроцессоре UltraSPARC с архитектурой SPARC-V9. Это будет полностью 64-разрядная операционная система, реализованная в соответствии со стандартом Unix System V Release 4.

По словам вице-президента Data General Джозеля Шварца, UltraSPARC — это подтверждение верного пути развития SPARC-систем. 64-разрядный процессор UltraSPARC — это первый процессор, разработанный в соответствии с сегодняшними требованиями поддержки распределенных технологий мультимедиа (например, этот процессор обеспечивает декомпрессию данных формата MPEG-2 в

реальном масштабе времени). Он содержит 5,2 миллиона транзисторов и производится с помощью технологии EPIC3 фирмы Texas Instruments. Архитектура SPARC-V9 поддерживает 64-разрядную адресацию данных, защиту от сбоев, быстрое переключение контекста, глубокую оптимизацию и 100-процентную бинарную совместимость конечных приложений.

24 мая 1995 года в Сан-Франциско на конференции SunWorld'95 было объявлено, что уже тринадцать независимых производителей компьютеров начали разработки на базе микропроцессора

UltraSPARC. Это Axil, CDAC, Cray, DTK, FORCE, Haitai Electronics, Hitron systems, LG Electronics, Sun, Tatung, Themis, Trigem, Toshiba.

**НАШИ
ТЕЗКИ**




... Через каждые несколько минут на меня обрушивался грохот стартующих неподалеку джетов. Впрочем, в Европе говорят не "джеты", а "аэропланы",

но "джет" лучше передает ощущение ударов по голове.

С. Лем. "Осмотр на месте"

Григорий Барон



Параллельные архитектуры серверов баз данных

Содержание

- Масштабируемость системы
 - Многопроцессорные системы
 - Гибкость архитектуры
- Производительность СУБД
 - Параллельные алгоритмы
 - Многопоточковая архитектура СУБД
- Смешанная загрузка в современных СУБД
 - Оптимизатор запросов
 - Управление ресурсами
 - Параллельная обработка запросов
- Постоянная доступность данных
 - Оперативное администрирование
 - Функциональная насыщенность СУБД
- Заключение
- Литература

Современный мир информации находится в постоянном движении и развитии. Главными силами в этом процессе являются растущие потребности заказчиков информационных систем и расширившиеся технологические возможности аппаратных и программных средств. Однако, несмотря на множество течений и концепций в области промышленной обработки данных, объектно-ориентирован-

ные и многомерные модели данных, реляционная концепция продолжает занимать центральное положение.

Одним из основных факторов развития является смещение информационных услуг в сторону обслуживания все более сложных запросов в реальном времени. Потребителям все труднее мириться с ограничениями существующих технологий, требующих искусственно разделять задачи массового ввода и модификации данных (OLTP), задачи анализа (DSS) и пакетную обработку (batch processing). Пока попытки совместить эти задачи в рамках одной вычислительной системы требуют огромных усилий по сопровождению системы. Смешанная загрузка определяется как оперативная обработка сложных транзакций (OLCP) и становится неотъемлемым качеством современных информационных систем.

Быстрое продвижение наблюдается в области аппаратных решений. Все более мощные вычислительные платформы становятся доступны по первому требованию при одновременном снижении стоимости аппаратных компонентов. Увеличилась пропускная способность сетей, продолжает снижаться стоимость оперативной памяти, постоянно растет емкость дисковых накопителей и скорость досту-

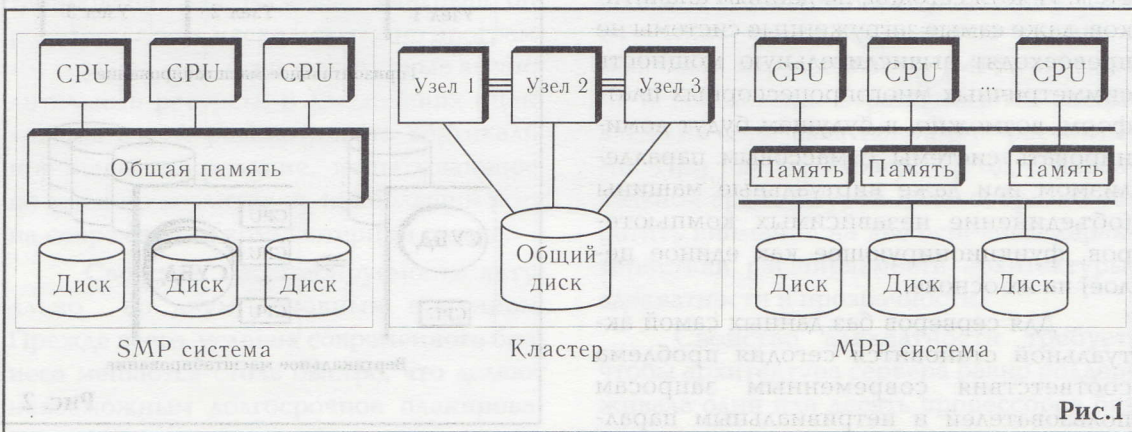


Рис.1

па к дисковым массивам. Однако наиболее важным изменением в компьютерных архитектурах можно считать использование множества процессоров для кардинального увеличения вычислительной мощности. Фактически, определились три архитектурных направления (рис. 1):

- Симметричные многопроцессорные системы (SMP) — наиболее часто используемая форма сильносвязанных многопроцессорных систем, т.е. систем, разделяющих единую оперативную память и, наиболее часто, — дисковую подсистему;

- Слабосвязанные многопроцессорные системы — совокупность самостоятельных компьютеров, объединенных в единую систему быстродействующей сетью и, возможно, имеющих общую дисковую подсистему, как, например, кластерные инсталляции;

- Системы с массовым параллелизмом (MPP) — системы с сотнями и даже тысячами процессоров. Детали их реализации могут значительно различаться.

Несмотря на перспективность каждого из перечисленных подходов, наиболее оптимальными с точки зрения стоимости и прозрачности наращивания можно считать симметричные многопроцессорные платформы. Добавление процессоров в них обходится относительно дешево, и при использовании соответствующей операционной среды не требует изменения программного обеспечения или принципов администрирования, причем, начиная уже с однопроцессорных систем. Для более дорогостоящих и ответственных систем необходимый уровень резервирования может быть достигнут с помощью кластеров, в т.ч. состоящих из SMP-систем. И хотя сегодня, по данным аналитиков, даже самые загруженные системы не превосходят вычислительную мощность симметричных многопроцессорных платформ, возможно, в будущем будут доминировать системы с массовым параллелизмом или даже виртуальные машины (объединение независимых компьютеров, функционирующее как единое целое) на их основе.

Для серверов баз данных самой актуальной становится сегодня проблема соответствия современным запросам пользователей и нетривиальным парал-

лельным архитектурам вычислительных платформ. Основной вопрос данной статьи — каким новым требованиям должны удовлетворять архитектура и функциональные возможности сервера баз данных в окружении параллельных аппаратных решений и ориентированных на параллелизм операционных систем. Для упрощения рассматривается реализация параллелизма на примере SMP-платформы — наиболее вероятной на сегодняшний день основы для построения крупного проекта. Автор намеренно не упоминает названия и точные возможности лидирующих на рынке продуктов — тема параллелизма настолько важна в производственной программе компаний Informix, Oracle и Sybase, что очень скоро возможности ведущих РСУБД в этой области будут отличаться лишь незначительными деталями в той же мере, насколько сейчас они имеют разные диалекты одного языка SQL.

Можно выделить четыре группы требований, определяющих с технической точки зрения потребительские качества современной СУБД:

- масштабируемость;
- производительность;
- возможность смешанной загрузки разными типами задач;
- обеспечение постоянной доступности данных.

В дальнейшем мы рассмотрим эти параметры подробнее.

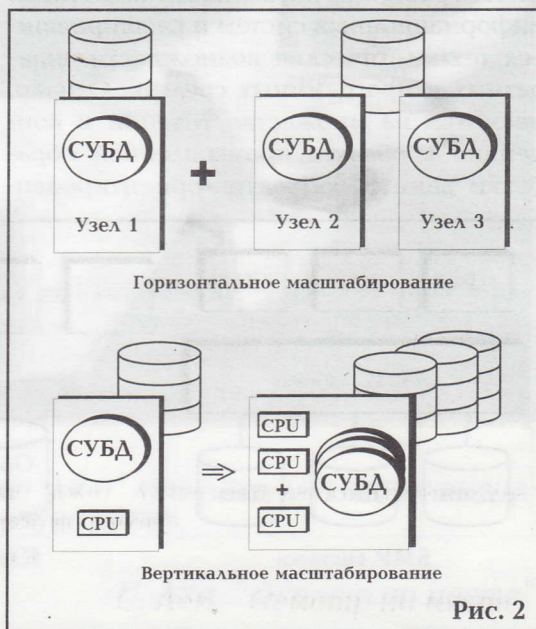


Рис. 2

Масштабируемость системы

Масштабируемость — такое свойство вычислительной системы, которое обеспечивает предсказуемый рост системных характеристик, например, числа поддерживаемых пользователей, скорости реакции, общей производительности и пр., при добавлении к ней вычислительных ресурсов. В случае сервера СУБД можно рассматривать два способа масштабирования — вертикальный и горизонтальный (рис. 2).

При горизонтальном масштабировании увеличивается число серверов СУБД, возможно, взаимодействующих друг с другом в прозрачном режиме, разделяя таким образом общую загрузку системы. Такое решение, видимо, будет все более популярным с ростом поддержки слабосвязанных архитектур и распределенных баз данных, однако обычно оно характеризуется сложным администрированием.

Вертикальное масштабирование подразумевает упрочнение отдельного сервера СУБД и достигается заменой аппаратного обеспечения (процессора, дисков) на более быстродействующее или добавлением дополнительных узлов. Хорошим примером может служить увеличение числа процессоров в симметричных многопроцессорных (SMP) платформах. При этом программное обеспечение сервера не должно изменяться (в частности, нельзя требовать закупки дополнительных модулей), так как это увеличило бы сложность администрирования и ухудшило предсказуемость поведения системы. Независимо от того, какой способ масштабирования использован, выигрыш определяется тем, насколько полно программы сервера используют доступные вычислительные ресурсы. В дальнейших оценках мы будем рассматривать вертикальное масштабирование, испытывающее, по мнению аналитиков, наибольший рост на современном компьютерном рынке.

Свойство масштабируемости актуально по двум основным причинам. Прежде всего, условия современного бизнеса меняются столь быстро, что делают невозможным долгосрочное планирова-

ние, требующее всестороннего и продолжительного анализа уже устаревших данных, даже для тех организаций, которые способны это себе позволить. Взамен приходит стратегия постепенного, шаг за шагом, наращивания мощности информационных систем. С другой стороны, изменения в технологии приводят к появлению все новых решений и снижению цен на аппаратное обеспечение, что потенциально делает архитектуру информационных систем более гибкой. Одновременно расширяется межоперабельность, открытость программных и аппаратных продуктов разных производителей, хотя пока их усилия, направленные на соответствие стандартам, согласованы лишь в узких секторах рынка. Без учета этих факторов потребитель не сможет воспользоваться преимуществами новых технологий, не замораживая средств, вложенных в недостаточно открытые или оказавшиеся бесперспективными технологии. В области хранения и обработки данных это требует, чтобы и СУБД, и сервер были масштабируемы. Сегодня ключевыми параметрами масштабируемости являются

- поддержка многопроцессорной обработки;
- гибкость архитектуры.

Многопроцессорные системы

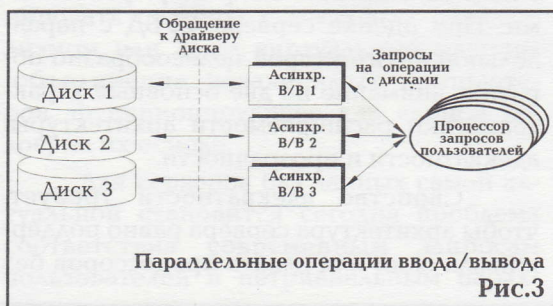
Для вертикального масштабирования все чаще используются симметричные многопроцессорные системы (SMP), поскольку в этом случае не требуется смены платформы, т.е. операционной системы, аппаратного обеспечения, а также навыков администрирования. С этой целью возможно также применение систем с массовым параллелизмом (MPP), но пока их использование ограничивается специальными задачами, например, расчетными. При оценке сервера СУБД с параллельной архитектурой целесообразно обратить внимание на две основные характеристики расширяемости архитектуры: адекватности и прозрачности.

Свойство адекватности требует, чтобы архитектура сервера равно поддерживала один или десять процессоров без

переустановки или существенных изменений в конфигурации, а также дополнительных программных модулей. Такая архитектура будет одинаково полезна и эффективна и в однопроцессорной системе, и, по мере роста сложности решаемых задач, на нескольких или даже на множестве (MPP) процессоров. В общем случае потребитель не должен дополнительно покупать и осваивать новые опции программного обеспечения.

Обеспечение прозрачности архитектуры сервера, в свою очередь, позволяет скрыть изменения конфигурации аппаратного обеспечения от приложений, т.е. гарантирует переносимость прикладных программных систем. В частности, в сильносвязанных многопроцессорных архитектурах приложение может взаимодействовать с сервером через сегмент разделяемой памяти, тогда как при использовании слабосвязанных многосерверных систем (кластеров) для этой цели может быть применен механизм сообщений. Приложение не должно учитывать возможности реализации аппаратной архитектуры — способы манипулирования данными и программный интерфейс доступа к базе данных обязаны оставаться одинаковыми и в равной степени эффективными.

Качественная поддержка многопроцессорной обработки требует от сервера баз данных способности самостоятельно планировать выполнение множества обслуживаемых запросов, что обеспечило бы наиболее полное разделение доступных вычислительных ресурсов между задачами сервера. Запросы могут обрабатываться последовательно несколькими задачами или разделяться на подзадачи, которые, в свою очередь, могут быть выполнены параллельно (рис. 3). Последнее более оптимально, поскольку правильная



реализация этого механизма обеспечивает выгоды, независимые от типов запросов и приложений. На эффективность обработки огромное воздействие оказывает уровень гранулярности рассматриваемых задач-планировщиком операций. При грубой гранулярности, например, на уровне отдельных SQL-запросов, разделение ресурсов вычислительной системы (процессоров, памяти, дисков) не будет оптимальным — задача будет простаивать, ожидая окончания необходимых для завершения SQL-запроса операций ввода/вывода, хотя бы в очереди к ней стояли другие запросы, требующие значительной вычислительной работы. При более тонкой гранулярности разделение ресурсов происходит даже внутри одного SQL-запроса, что еще нагляднее проявляется при параллельной обработке нескольких запросов. Применение планировщика обеспечивает привлечение больших ресурсов системы к решению собственно задач обслуживания базы данных и минимизирует простои.

Гибкость архитектуры

Независимо от степени мобильности, поддержки стандартов, параллелизма и других полезных качеств, производительность СУБД, имеющей ощутимые встроенные архитектурные ограничения, не может наращиваться свободно. Наличие документированных или практических ограничений на число и размеры объектов базы данных и буферов памяти, количество одновременных подключений, на глубину рекурсии вызова процедур и подчиненных запросов (subqueries) или срабатывания триггеров базы данных является таким же ограничением применимости СУБД как, например, невозможность переноса на несколько вычислительных платформ. Параметры, ограничивающие сложность запросов к базе данных, в особенности, размеры динамических буферов и стека для рекурсивных вызовов, должны настраиваться в динамике и не требовать остановки системы для реконфигурации. Нет смысла покупать новый мощный сервер, если ожидания не

могут быть удовлетворены из-за внутренних ограничений СУБД.

Обычно узким местом является невозможность динамической подстройки характеристик программ сервера баз данных. Способность на ходу определять такие параметры, как объем потребляемой памяти, число занятых процессоров, количество параллельных потоков выполнения заданий (будь то настоящие потоки (threads), процессы операционной системы или виртуальные процессоры) и количество фрагментов таблиц и индексов баз данных, а также их распределение по физическим дискам БЕЗ останова и перезапуска системы является требованием, вытекающим из сути современных предложений. В идеальном варианте каждый из этих параметров можно было бы изменить динамически в заданных для конкретного пользователя пределах.

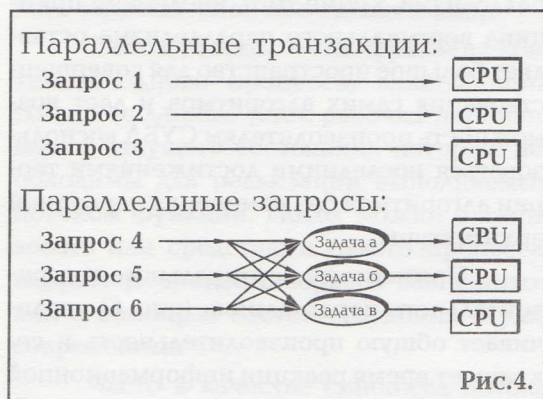
Производительность СУБД

Среди многих факторов, влияющих на производительность СУБД, выберем два, имеющие прямое отношение к параллельным вычислительным платформам сегодняшнего дня:

- поддержка параллелизма
- реализация многопоточковой архитектуры

Параллельные алгоритмы

Одним из способов достижения более высокой производительности является использование алгоритмов распараллеливания заданий. В СУБД существует

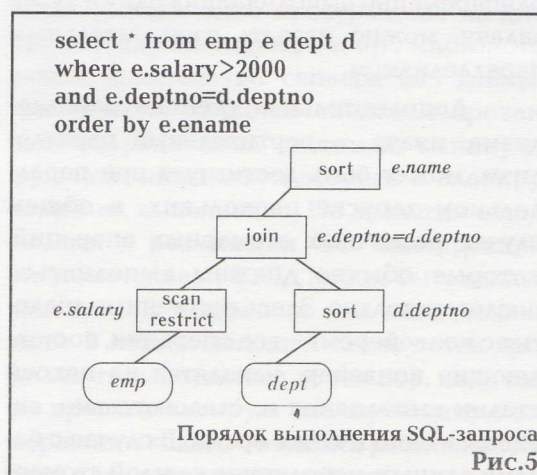


три области применения таких алгоритмов:

- параллельный ввод/вывод,
- параллельные средства и утилиты администрирования,
- параллельная обработка запросов к базе данных.

Распараллеливание ввода/вывода (рис. 4) в сочетании с оптимальным планированием заданий позволяет осуществить весьма эффективный одновременный доступ к фрагментированным таблицам и индексам, расположенным на нескольких физических дисках, тем самым многократно ускоряя операции с относительно медленными внешними устройствами. Выполнение административных утилит (в том числе реорганизация данных, сортировка, построение индексов, загрузка и выгрузка данных, архивирование и восстановление баз данных) также должно быть полностью параллельным, включая распараллеливание операций ввода/вывода, хотя на практике это требует доработки лишь небольшого числа механизмов.

В отличие от параллельных ввода/вывода и администрирования, параллелизм в обработке запросов реализуется гораздо сложнее. Теоретическим обоснованием возможности распараллеливания запросов в реляционной СУБД является свойство реляционной замкнутости — результатом каждого реляционного оператора: SELECT — выборка подмножества строк отношения (таблицы); PROJECT — выборка подмножества полей (столбцов); JOIN — соединение двух таблиц, является



новое отношение, а поскольку любой запрос можно разбить на иерархию элементарных операторов, то разумно попытаться выполнить их параллельно. Языку запросов SQL, несомненно, внутренне присущ параллелизм. Обработка запроса обычно состоит из множества атомарных операций, состав и последовательность которых, т.е. план запроса (рис. 5), определяется оптимизатором после просмотра нескольких вариантов. В недалеком прошлом все эти операции выполнялись в строгой последовательности, результат каждой служил исходным набором данных для последующей. Параллелизм в обработке запросов подобен идее "разделяй и властвуй" или коллективному выполнению задания. Вследствие деления задания на части между несколькими работниками результат достигается быстрее. Для обеспечения параллелизма в сервере баз данных производитель должен выбрать такой набор атомарных операций, например, scan — просмотр таблицы, sort — сортировка, join — соединение двух таблиц, которые (а) могли бы тиражироваться и выполнять свою часть задачи одновременно, и (б) результаты их работы можно было бы объединить, как если бы задача была выполнена одним исполнителем.

Для обеспечения параллелизма в сервере СУБД требуется, чтобы декомпозиция и параллельное выполнение запроса были осуществимы независимо от средств межзадачного взаимодействия, принятых в конкретной вычислительной системе. Создание множества тиражируемых копий одной атомарной операции, одновременно выполняющих одну и ту же задачу, можно назвать горизонтальным параллелизмом.

Дополнительная степень параллелизма, иначе — вертикальный параллелизм, может быть достигнута при параллельном запуске нескольких, в общем случае различных атомарных операций, которые обычно должны выполняться последовательно. Здесь применима аналогия с конвейером — все операции, составляющие конвейер, находятся на некоей стадии выполнения и, следовательно, активны в одно и то же время. В случае с базами данных, исполнение каждой атомар-

ной операции начинается сразу же с появлением данных, которые она должна обрабатывать, не дожидаясь завершения предыдущего шага и накопления полученных там данных в промежуточном буфере. Сигналом к запуску последующей операции является момент начала поступления данных для нее. Например, операция соединения таблиц может начинать работать одновременно с операциями последовательного чтения с диска и фильтрации строк, а ее результаты по мере по-



ступления могут быть переданы на вход операции сортировки, например, для начального разделения данных по ключу в соответствии с алгоритмом упорядочивания, и при этом все три вида операций: сканирования, объединения и сортировки, продолжают работать одновременно. Такой подход, во многом сходный с конвейерной обработкой в современных процессорах, носит название "потока данных" (data flow) или "управления по требованию" (demand driven). Дополнительным плюсом этого подхода является то, что его можно успешно применять в аппаратной архитектуре с любой степенью параллелизма, кроме того, внедрение принципа вертикального параллелизма оставляет большое пространство для совершенствования самих алгоритмов и дает возможность производителям СУБД воспользоваться последними достижениями теории алгоритмов без изменения серверной архитектуры.

Применение вертикального и горизонтального параллелизма (рис. 6) увеличивает общую производительность и сокращает время реакции информационной системы за счет ускорения процедур об-



работки данных. В свою очередь, рост объемов обрабатываемых данных обеспечивается вертикальным и/или горизонтальным масштабированием аппаратной платформы.

Многопоточковая архитектура СУБД

Идея обеспечить распределение вычислительных ресурсов между запросами от многих пользователей непосредственно в сервере СУБД, не доверяя средствам операционной системы, возникла из необходимости обслуживать все большее число пользователей, снизив при этом значительные накладные расходы ОС на сохранение и загрузку контекста их процессов, по сути подобных друг другу. Способ организации одновременной обработки множества запросов внутри сервера баз данных при минимальных накладных расходах на переключение контекста и максимальном разделении вычислительных ресурсов между ними называется истинно многопоточковой (рис. 7) архитектурой (не следует путать с имитацией многопоточковости с помощью дополнительного уровня между клиентами и сервером — диспетчера-мультиплексора запросов, что позволяет серверу обрабатывать запросы лишь последовательно).

Поток (thread) — это фрагмент контекста одного процесса, включающий только те данные (стек рабочих переменных и текущего состояния), которые необходимы для реализации выполняемых потоком функций. Поток можно организовать или средствами самого процесса, например, процесса сервера баз данных, или с помощью соответствующих служб современных ОС.

Часто в качестве синонима потока используется термин "легковесный про-

цесс" (light-weighted process), поскольку затраты ресурсов на запуск, останов, а также создание или уничтожение потока силами управляющего процесса гораздо ниже по сравнению с действиями ОС над обычными процессами. В принципе, для обеспечения параллелизма поток может быть клонирован или поделен на подзадачи — потоки более низкого уровня, примерно так же, как и любой процесс внутри операционной системы. В результате потоки избавляют операционную систему от создания, планирования и манипуляции множеством процессов, и, следовательно, задачи самой ОС существенно меньше загружают вычислительную систему.

Итак, истинно многопоточковая архитектура обеспечивает более совершенное разделение ресурсов между прикладными задачами и операционной системой, а также более высокий уровень стабильности по мере роста числа пользователей, чем традиционные многопроцессные архитектуры. Но, помимо этого, многопоточковость может повысить независимость между модулями сервера баз данных, поскольку выполнение условных переходов внутри сервера может основываться на событиях, возникающих в результате взаимодействия потоков, а не на прямой реализации кода, что позволяет надеяться на повышение стабильности и безболезненное развитие кода самого сервера.

Выполняя все функции управления ресурсами, необходимыми СУБД, включая динамическое размещение буферов в памяти, выделение пространства на диске, систему блокировок и пр., многопоточковая архитектура сервера баз данных фактически является специализированной операционной системой, манипулирующей множеством потоков и планирующей их выполнение. По аналогии с подходами в реализации ОС, планировщик многопоточковой СУБД может быть выполнен как вытесняющий или невытесняющий. В невытесняющей схеме поток продолжает выполняться до тех пор, пока сам не просигнализирует планировщику о завершении какого-то кванта работы, после чего планировщик назначает на выполне-

ние другой поток. Обычно такой планировщик использует кольцевой (round-robin) алгоритм постановки потоков в очередь на запуск. Выполняемый поток определяет момент своей остановки по истечению временного интервала, выполнению критической секции кода или по необходимости выполнить блокирующий вызов, например, операцию ввода/вывода. Такой механизм удобен для сервера, занимающегося исключительно обслуживанием базы данных, или в среде однотипных запросов, где потоков относительно немного и, следовательно, их поведение хорошо предсказуемо. В вытесняющей схеме поток может быть прерван по инициативе планировщика, например, при попадании в очередь более приоритетного потока. Этот принцип эффективен в системах с неспецифицированными запросами или при параллельной загрузке системы задачами, не связанными с базами данных, т.е. там, где стратегия планирования в зависимости от решаемых задач может изменяться в широких пределах.

Смешанная загрузка в современных СУБД

Эволюция в области информационных систем приобретает все более отчетливую направленность на объединение трех видов задач: оперативной обработки транзакций (OLTP), поддержки принятия решений (DSS) и пакетной обработки (batch processing), которые ранее искусственно отделялись. Одновременное исполнение задач смешанного характера, разделяющих общие вычислительные ресурсы и базы данных, носит название "оперативная обработка сложных транзакций" (OLCP). Управление ресурсами, оптимизация и администрирование в гибридной системе гораздо сложнее, чем в системе, ориентированной на решение единственной задачи, поскольку различные типы приложений предъявляют противоречивые требования к конфигурации. Тем не менее, преимуществом концепции смешанной загрузки является более полное соответствие задачам реальной жизни, а не удобству обработки данных. Например, экспертный анализ или загрузку/выгрузку данных пользователям хоте-

лось бы производить в реальном времени, одновременно с вводом и модификацией данных. С применением параллельных архитектур, значительно ускоряющих обработку и обеспечивающих масштабируемость, ожидается появление систем, поддерживающих смешанную загрузку.

К сожалению, путь к реализации смешанной загрузки тернист. Часто производители СУБД предлагают архитектурные решения сервера, специфичные для определенных классов приложений. Например, сервер для обработки транзакций может строиться в предположении, что

- наиболее часто используются короткие транзакции,
- обычно транзакции не используют одинаковые данные,
- операторы затрагивают обычно только несколько строк,
- только несколько таблиц имеют большие размеры или могут быть значительно изменены.

Реализация такого сервера опирается на физические методики сокращения операций с дисками, обработку небольших объемов данных в памяти, примитивный оптимизатор запросов, а также требования к приложениям — исключить конкуренцию запросов в использовании ресурсов и данных.

Для сервера системы принятия решений и оперативного анализа данных выдвигаются следующие требования:

- объем таблиц в целом не ограничен,
- часто требуется просмотр всех строк таблицы,
- число таблиц, участвующих в транзакции, может быть велико,
- обычно транзакции не модифицируют данные,
- высока вероятность разделения ресурсов и данных разными задачами.

Как следствие, необходимы мощный оптимизатор, эффективные методики сканирования и соединения таблиц, а также совершенные механизмы обеспечения параллельного доступа к данным нескольких приложений.

Сервер, ориентированный на пакетную обработку, полагается на истинность следующих предположений:

- обычно большие и сверхбольшие размеры таблиц,
- продолжительность транзакций может быть велика,
- обычно требуется просмотр и/или модификация всех строк таблицы,
- одновременный запуск очень небольшого числа задач,
- характерны высокая загрузка процессора и заполнение оперативной памяти.

Помимо прочих, в режиме пакетной обработки выполняются административные операции, например, по реорганизации базы данных. Все это предъявляет требования к наименьшей избыточности базовых алгоритмов, определяющих скорость примитивных операций. В условиях смешанной загрузки производительность таких СУБД будет крайне низка, либо потребуются неприемлемо высокий уровень сопровождения баз данных и приложений.

Основные факторы, вовлеченные в эффективную реализацию универсальной системы, можно определить как:

- оптимизация запросов;
- эффективное управление ресурсами;
- параллельная обработка запросов.

Оптимизатор запросов

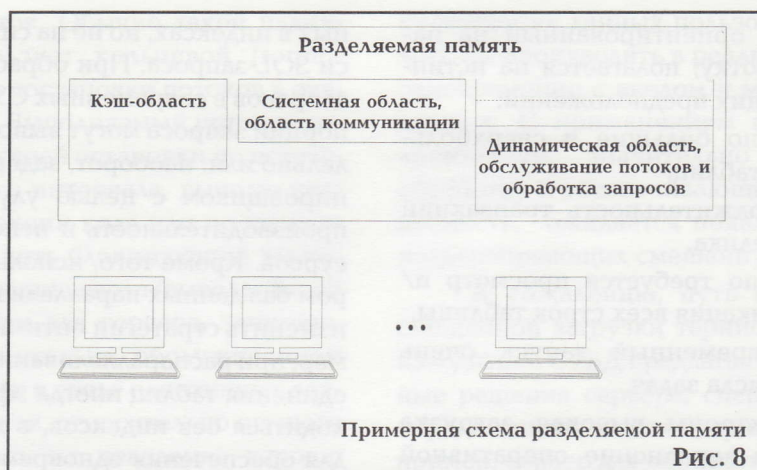
Возможности оптимизатора запросов в значительной мере определяют способности сервера эффективно обрабатывать SQL-операторы, затрагивающие несколько таблиц и множество строк. В частности, оптимизатор выбирает из нескольких вариантов оптимальный план выполнения запроса (рис. 5), в котором в виде дерева определены порядок перебора индексов и тип соединения таблиц на каждом шаге выполнения. Относительная стоимость запроса оценивается, исходя из предполагаемого количества операций с диском и операций сравнения в памяти. Оптимизатор должен строить свои оценки на статистике и на распределении дан-

ных в индексах, но не на синтаксисе записи SQL-запроса. При обработке сложных запросов в современных СУБД различные порции запроса могут выполняться параллельно или, наоборот, задерживаться планировщиком с целью улучшить общую производительность и использование ресурсов. Кроме того, используемый сервером баз данных параллелизм может резко изменить стратегии оптимизации. Например, при распараллеливании операции соединения таблиц иногда эффективнее обходиться без индексов, в других случаях для обеспечения одновременности доступа к данным необходимо тщательно учитывать уровень изоляции транзакций. Поэтому здесь необходима тесная кооперация между оптимизатором и подсистемой манипуляции данными, учитывающая все тонкости последующей обработки запроса в многопользовательском окружении.

Управление ресурсами

Оптимальное управление ресурсами обеспечивает поддержку прозрачности доступа к ресурсам в целом и эффективное использования каждого ресурса в отдельности. Поддержка прозрачности доступа очень актуальна. Например, создавая приложение в архитектуре клиент/сервер, разработчик не может и не должен строить предположений о том, как на самом деле взаимодействуют клиент и сервер. Использование в качестве среды коммуникаций сети или сегмента разделяемой памяти, никак не должно влиять на идеи разработки, точно так же приложение не может как-то ограничивать коммуникационные решения СУБД. Если приложение и сервер баз данных находятся на одном компьютере, то пересылку сообщений между ними наиболее эффективным образом можно организовать через разделяемую память. Это может быть особенно эффективно для просмотра результатов обширной выборки с помощью курсора.

Одним из наиболее важных ресурсов является оперативная память. Оптимальное управление распределением памяти настолько же критично для производительности системы, как и уменьшение загрузки ОС, требующее многопоточной архитектуры сервера баз данных. Прило-



жения СУБД используют разделяемую память для сохранения контекста подключения, а также кэширования запросов и результатов. В некоторых серверах баз данных память отводится отдельно для каждого пользователя, и это значит, что считанные в нее страницы данных или индексов (единицы обмена СУБД с диском) будут недоступны другим пользователям до тех пор, пока они не считают эти данные повторно. В других архитектурах в память помещаются большие объемы информации о возможном откате транзакций (undo log) каждого пользователя, что противоречит используемому в многопоточной архитектуре принципу минимальности хранимого контекста подключения, необходимого для эффективного управления потоками. Необходимы продуманные алгоритмы управления памятью, например, динамические разделяемые всеми приложениями буфера и кэш-области, наращиваемое по требованию распределение стека, функциональное разделение структур в памяти в соответствии с типами решаемых задач (рис. 8). Без этого архитектура сервера баз данных будет неспособна одновременно удовлетворить потребностям оперативной обработки транзакций, систем принятия решений и пакетной обработки.

Следующий момент, который стоит отметить, — исключение простаивания ресурсов процессоров. При необходимости выполнить вызов, блокирующий процесс, например, операцию ввода/вывода, есть смысл передать этот вызов как запрос другому процессу, а вместо ожидающего окончания операции потока собствен-

ный планировщик сервера баз данных может запустить следующий поток из очереди на обработку. При этом процесс и, следовательно, обработка запросов не будут принудительно остановлены средствами ОС. В деятельности планировщика заданий внутри сервера важна также granularity той порции обработки, которая по каким-либо причинам, например, для лучшей изоляции транзакции, должна завершиться до передачи управления другому потоку. Обычно квант обработки имеет некий функциональный смысл — завершение SQL-запроса или атомарной операции. Если операция оказывается продолжительной, например, при соединении таблиц в сложном аналитическом запросе, то это может вызвать общую задержку системы, и совершенно недопустимо при совмещении задач принятия решений с чувствительными к времени реакции задачами обработки транзакций. В условиях смешанной загрузки предпочтительно использование серверов, поддерживающих достаточно стабильный размер кванта обработки.

Параллельная обработка запросов

Параллельная обработка является решением при низком быстродействии, характерном для сложных запросов, и больших размерах обрабатываемой информации, например, массивов обновляемых крупных таблиц. За счет совмещения обработки небольших порций данных, параллелизм значительно улучшает производительность систем принятия решений и пакетной обработки. Эти

улучшения позволяют включать сложные запросы в массовые транзакции, не жертвуя целостностью данных и изолированностью каждой транзакции, что было бы невозможным в традиционных системах. Имея приемлемое время реакции, аналитические (DSS) запросы не конкурируют с оперативной обработкой транзакций (OLTP) и не разрушают среду OLTP.

Постоянная доступность данных

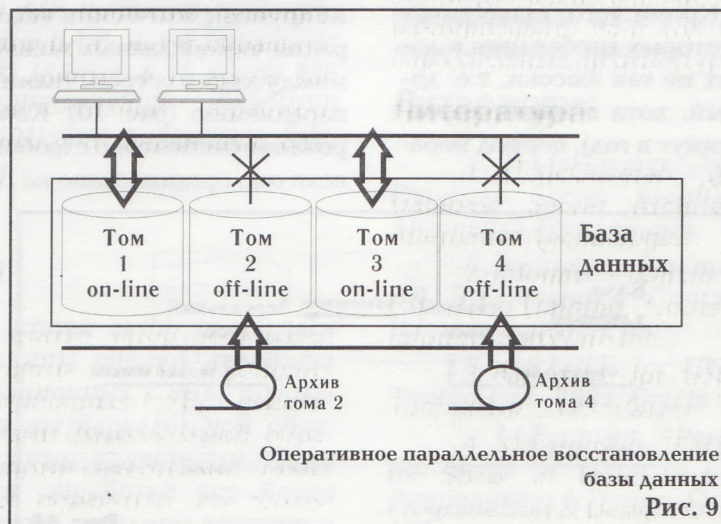
Основные моменты, обеспечивающие постоянную готовность современных СУБД, можно определить как:

- оперативное администрирование;
- функциональная насыщенность СУБД.

Оперативное администрирование

Средства администрирования в идеале призваны поддерживать бесперебойное функционирование СУБД, что подразумевает сведение к минимуму планируемых или сбойных простоев системы. На практике систему иногда требуется остановить ради выполнения какой-либо утилиты. Утилиты для пакетной загрузки/выгрузки данных, архивирования и восстановления, проверки целостности, реорганизации индекса и пр. — все должны эффективно выполняться в оперативном (on-line) режиме, без остановки СУБД, причем для ускорения предпочтительно использование параллельных алго-

ритмов. При возникновении сбоя во время этих, как правило, продолжительных операций, с помощью контрольных точек (checkpoints) или даже специальных журналов должен обеспечиваться повторный запуск административных утилит непосредственно с точки останова. Для сокращения времени автономного (off-line) обслуживания необходима поддержка таких возможностей, как автоматическое, без участия оператора, оперативное архивирование заполненных журналов и базы данных одновременно на несколько архивных устройств, автоматический перезапуск системы с контролируемым временем восстановления и т.д. Задачи администрирования, включая мониторинг и реконфигурацию ресурсов системы, часто вступают в противоречие с доступностью данных. В идеале, выполнение операций по реорганизации и перемещению крупных таблиц и индексов в пределах системы не должно ограничивать доступ к таблице, части таблицы или к базе данных. Даже в случае необходимости выключения части базы данных, например, в силу физического повреждения таблиц или их частей, работоспособные части таблиц и базы данных должны оставаться доступными для приложений в течение времени восстановления (рис. 9). При этом приложения должны, конечно, знать, что вместо целой таблицы для запросов доступна лишь ее часть. Вообще говоря, любые административные действия, включая физическое планирование, размещение, перемещение и реорганизацию баз данных,



управление дисковым пространством, архивирование и восстановление журналов и частей баз данных, перезапуск СУБД должны быть максимально прозрачными для приложений. Средства мониторинга серверов СУБД должны позволять пользователям строить свои собственные управляющие процедуры, отражающие специфику применения СУБД. При этом важны такие параметры, как возможность централизации управления, подобно средствам прежних централизованных систем (mainframes), выполнение операций в назначенное время без участия оператора (unattended or scheduled), одновременное использование всех доступных архивных устройств вычислительной платформы. Все это обеспечивает максимальную готовность и доступность информационной системы.

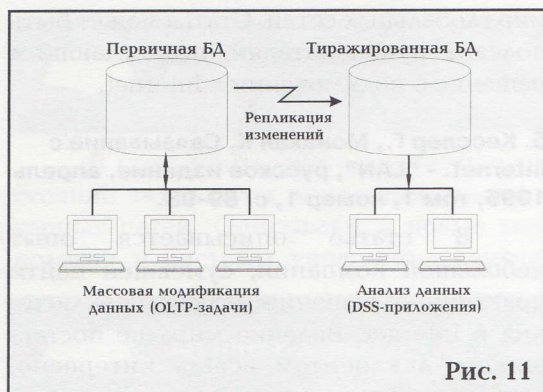
Функциональная насыщенность СУБД

Общим термином "функциональная насыщенность" можно определить множество механизмов, используемых серверами баз данных (а) для уменьшения последствий системных сбоев и (б) для повышения прозрачности доступа к дублированным данным при нормальном функционировании. Теоретически СУБД обязана обеспечивать доступ к данным по чтению и записи, независимо от обстоятельств, будь то сбой аппаратной платформы или какого-то ее компонента. Для обозначения такого уровня доступности используется термин "нечувствительность к сбоям". Кроме того, существует класс систем, в которых требования к доступности данных не так высоки, т.е. допустим временный, хотя и очень короткий (несколько минут в год), период неработоспособности.

Системы, обеспечивающие непрерывный доступ к данным (fault tolerant) или почти непрерывный (high availability) обычно опираются на различные формы избыточности. Как правило, это системы дублирования аппаратного обеспечения и контролируемой избыточности данных. Аппаратная избыточность может включать платформы с полным резервированием, поддерживающие (standby) процессоры, диски с двойным интерфейсом (dual-port), дисковые массивы и пр. Один из вариантов — зеркалирование дисков, когда один диск используется в качестве копии другого и может быть использован при сбое вместо него. Хотя аппаратная избыточность и важна для повышения общей надежности системы, ее реализация, как правило, не ориентирована на обработку транзакций СУБД и на связанные с этим специфические ограничения, например, обеспечение атомарности транзакции. В результате СУБД не может воспользоваться преимуществами чисто аппаратных решений резервирования системы для повышения своей производительности. Управляемая избыточность данных обычно представлена в двух формах — программное зеркалирование (software mirroring) и тиражирование (replication) данных. Программное зеркалирование дисков, называемое также дуплексированием (duplexing) или мультиплексированием (multiplexing), может не только защитить от аппаратных сбоев, но и улучшить производительность. Поскольку зеркалирование баз данных (или ее частей — таблиц(ы), индексов, их фрагментов и пр.) производится на другом физическом устройстве, то операции чтения данных можно распределить между двумя устройствами и производить параллельно (рис. 10). Конечно, зеркалирование бесполезно с любой точки зрения, если оно организовано на одном диске.



Рис. 10



В случае повреждения зеркалируемого диска все операции автоматически переносятся на исправный диск, сбойный диск выводится в отключенное состояние, причем приложения не замечают каких-либо изменений в конфигурации системы. После замены неисправного диска параллельно с работой пользователей запускается процесс оперативной синхронизации зеркальных дисков (on-line mirroring), на физическом уровне копирующий рабочий диск.

Тиражирование в системах, требующих в первую очередь повышенной надежности, в целом подобно зеркалированию, но здесь копия данных может поддерживаться удаленно. Если происходит копирование всей базы данных, то обычно это делается с целью обеспечить горячий резерв (warm standby). Однако в некоторых реализациях есть возможность использовать копию для просмотра (без модификации) данных. Это способно обеспечить значительные преимущества для систем со смешанной нагрузкой, поскольку приложения для принятия решений, генерации отчетов и т.п. могут обращаться к копии базы данных, в то время как приложения оперативной обработки транзакций используют первичную базу данных (рис. 11).

Заключение

В данной статье автор постарался изложить принципы оценки новейших реализаций реляционных СУБД, использующих механизмы параллельной обработки. Важнейшими качествами таких СУБД являются обработка все более сложных запросов в реальном времени и

управление большими и сверхбольшими базами данных (VLDB), что выдвигает параллельные СУБД в отдельный класс инструментальных информационных систем, обслуживающих наиболее ответственные потребности промышленной обработки данных.

Надо отметить, что разработчики последнего поколения РСУБД явились пионерами в промышленном освоении вычислительной мощности многопроцессорных архитектур. В качестве аппаратной платформы для демонстрации возможностей параллельных СУБД в статье рассматривается наиболее практически освоенная и широко распространенная симметричная многопроцессорная (SMP) архитектура. В будущем планируется оценить слабосвязанные многопроцессорные архитектуры.

В качестве основных параметров, характеризующих потребительские свойства параллельных СУБД, были выбраны

- масштабируемость;
- производительность;
- оперативная обработка сложных запросов;
- оперативное администрирование;

т.е. задачи, наиболее полно реализуемые средствами параллельной обработки. Автор намеренно не пытался ограничиться текущим состоянием какого-либо конкретного продукта — уже в течение ближайшего года возможности серверов Informix, Oracle и Sybase в целом будут очень близки, что позволит характеризовать 95-й год как начало эры параллельных архитектур.

Литература:

1. D.McGoveran. "An Evaluation of Database Server Architectures" 1993, Alternative Technologies.
2. Informix — OnLine Dynamic Server 6.0, 7.1. Training Course 1993, 1994, Informix Software Inc.
3. ORACLE for UNIX. Performance Tuning Tips. 1993, Oracle Corp.
4. M.Ferguson. "Parallel Database — the Shape of Thing to Come". Database Programming & Design, October, November 1994.

ЧТО ЕЩЕ ПОЧИТАТЬ



1. Митчел Дж. и др. Система Spring - общий обзор. - "Открытые системы", 1(9) 1995, с. 40-49.

Операционные системы на пороге кардинальных изменений. Время динозавров, шатающихся под собственной тяжестью, постепенно уходит. Новые идеи, в основе которых лежит концепция микроядра, необычайно красивы. Чтение обзора системы Spring позволяет заглянуть в будущее, которое готовится уже сейчас.

2. Добрецов А. Sendmail. - "Открытые системы", 1(9) 1995, с. 72-77.

Электронная почта — один из самых употребительных сервисов, но ее конфигурирование часто вызывает затруднения. Статья А. Добрецова способна помочь начинающим администраторам Unix-систем преодолеть стоящие перед ними проблемы.

3. Харрингтон М. Mosaic: распахнутая дверь в интерактивный мир. - "Открытые системы", 1(9) 1995, с. 67-71.

Ваша организация еще не имеет выход в Internet? Возможно, статья Харрингтона заставит Вас начать действовать. В ней сжато и в то же время очень точно описаны предоставляемые Internet информационные сервисы.

4. Шелтцер А. Азбука Internet. - "LAN", русское издание, апрель 1995, том 1, номер 1, с. 83-87.

В журнальной аннотации статья названа "кратким курсом ликбеза". Действительно, в ней излагаются основы знаний, которыми должен обладать современный специалист, входящий в

мир глобальных сетей. Статья может быть полезна руководителям, принимающим решение о подключении к Internet.

5. Кесслер Г., Монахан К. Связывание с Internet. - "LAN", русское издание, апрель 1995, том 1, номер 1, с. 89-93.

В статье описывается опыт небольшой компании, сумевшей найти практическое решение задачи подключения к Internet. Видение мира не поставщиком, а клиентом всегда интересно, особенно когда клиент активно и грамотно ищет способы приобщиться к последним достижениям информатики.

6. Продукты года. - "LAN", русское издание, апрель 1995, том 1, номер 1, с. 6-25.

Приводимый обзор сетевых продуктов года позволяет понять, насколько многогранным является понятие корпоративной сети. Сам подбор продуктов носит весьма субъективный характер и охватывает в основном дешевый конец рыночного спектра. Тем не менее, парад звезд даже местного масштаба — это всегда интересно. Кроме того, обзор позволяет руководителям и специалистам скорректировать свои системы отчета и яснее понять, что и за какие деньги сейчас можно купить.

7. Штейнке С. Как живешь, Ethernet? - "LAN", русское издание, апрель 1995, том 1, номер 1, с. 26-31.

В статье дается представление об основных ресурсах повышения производительности локальных сетей Ethernet. Имеются в виду коммутация Ethernet и технология быстрого Ethernet. Приводятся топологические правила сетей Ethernet, разъясняется смысл чисел, фигурирующих в связи с Ethernet. В этом отношении статья уникальна — приводимые в ней таблицы и рисунки могут послужить существенным подспорьем при проектировании реальных сетей масштаба предприятия.



Jet Info т. 1

Современная технология компьютерных сетей

В статье показано современное состояние технологии локальных компьютерных сетей. Описываются новые технологии и системы управления сетями.

Речь идет о структурированных кабельных сетях, коммутаторах Ethernet, быстрой Ethernet, ATM и о наиболее перспективных системах сетевого управления.



Jet Info т.2, вып. 1

Интеллектуальная СУБД "Ingres"

В статье описаны структура и основные возможности версии 6.4 интеллектуальной реляционной СУБД

Ingres. Изложение новых возможностей и механизмов сопровождается подробными комментариями.



Jet Info т.2, вып.2

Современные технологии СУБД

Статья посвящена изменению роли реляционных СУБД в структуре современной организации. Описываются тре-

бования к СУБД нового поколения. В качестве примера СУБД, удовлетворяющей этим требованиям, приводится Ingres.



Jet Info т.2, вып.3

Ingres в действии

В кратком обзоре, посвященном СУБД Ingres, приводятся сведения о компании ASK Group и ее стратегии,

рассматриваются примеры успешного использования СУБД Ingres крупнейшими компаниями.



Jet Info т.3

Надежность, готовность и обслуживаемость в системах SPARCcenter 2000 и SPARCserver 1000

В статье разъясняются понятия надежности, готовности и обслуживаемости, описываются средства обеспечения надежности, готовности и обслужи-

ваемости, начиная от спецификаций микросхем и кончая механизмами реконфигурации и избыточности.

Информационный бюллетень *Jet Info*

Зарегистрирован Комитетом РФ по печати 20 июля 1995 года, № 013951
Индекс по каталогу РОСПЕЧАТИ - 32555

Главный редактор
В.А.Галатенко

Технический редактор
С.И.Демочкин

103006, Москва
Краснопролетарская ул., 6
Тел.: (095) 972 11 82, 972 13 32
Факс:(095) 972 07 91

E-mail: info@jet.msk.su

© Jet Infosystems 1995

Полное или частичное воспроизведение материалов, содержащихся в настоящем издании, допускается только по согласованию с Jet Infosystems.

Перед Вами один из номеров бесплатного периодического информационно-технического бюллетеня "Jet Info", издаваемого компанией "Инфосистемы Джет" - ведущим российским системным интегратором в области UNIX-систем.

Если Вы еще не являетесь подписчиком "Jet Info", но хотели бы получать его регулярно, просим Вас отправить в наш адрес заполненный купон этого объявления. Отправьте этот купон и в том случае, если Вы сменили свой почтовый адрес или решили изменить форму доставки Вам бюллетеня.

Информация для подписчиков: не забудьте в течение каждого декабря и июля обязательно отправлять нам этот отрывной купон для перерегистрации.



1. Организация _____

2. Фамилия, имя, отчество и должность лица, в чей адрес будет осуществляться рассылка _____

3. Индекс и почтовый адрес _____

4. Телефон и/или факс _____

5. Адрес электронной почты _____

6. В какой форме Вы хотели бы получать бюллетень

печатный вариант по почте

электронную версию по электронной почте

7. Прежняя форма и адрес подписки (для подписчиков, меняющих адрес и форму подписки, либо тех, кто проходит перерегистрацию). _____

8. Общее число компьютеров в организации _____

9. Ваша организация использует

Вычислительную технику

Apple

DEC Alpha

DEC VAX

IBM PC

HP 9000

Silicon Graphics

Sun Microsystems

другое _____

Операционные системы

MS-DOS (Windows)

Novell NetWare

SCO Unix

Solaris / SunOS

UnixWare

VAX VMS

другое _____

10. Материалы на какую тему заинтересовали бы Вас в первую очередь _____

Мы будем рады, если кто-то из Вас захочет принять участие в подготовке выпусков "Jet Info" в качестве автора. Будем также признательны за любые конструктивные замечания и предложения по всем аспектам подготовки, выпуска и распространения нашего бюллетеня.

Редакция "Jet Info"