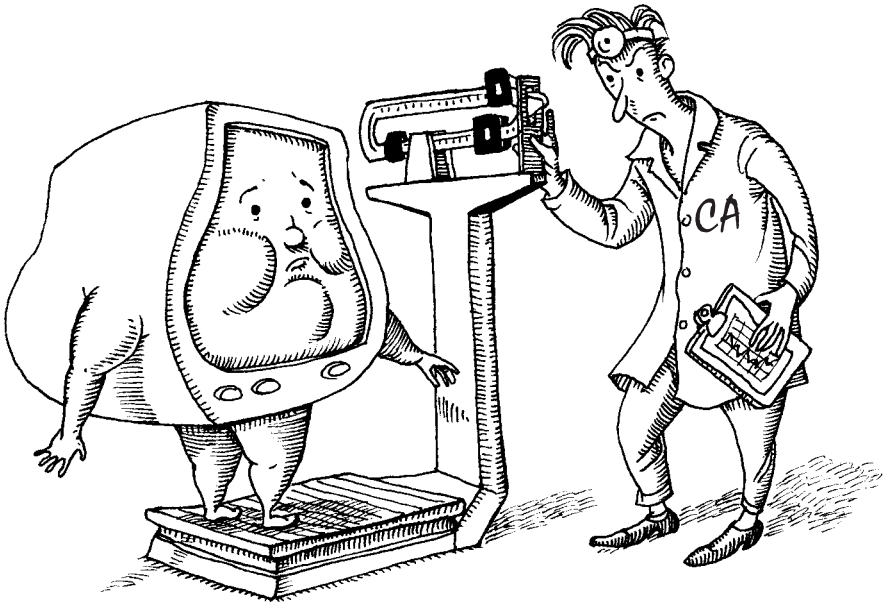
Глава 29

Анализ производительности



Анализ производительности и ее настройку часто относят к магической стороне системного администрирования. Конечно же, магии здесь никакой нет, но можно говорить о симбиозе науки и искусства. “Научная” часть включает тщательное проведение количественных измерений и применение научного подхода. Составляющая “искусства” связана с необходимостью сохранять баланс ресурсов на практическом уровне, поскольку оптимизация для одного приложения или пользователя может отрицательно повлиять на другие приложения или на других пользователей. Как это часто бывает в жизни, невозможно сделать счастливыми всех сразу.

В блогосфере бытует мнение, что сегодня проблемы производительности кардинально отличаются от аналогичных проблем предыдущих десятилетий. Но это не совсем так. Да, системы стали гораздо сложнее, но определяющие факторы производительности и высокоуровневые абстракции, используемые для ее измерения и управления, не меняются. К сожалению, повышение производительности систем сильно коррелирует со способностью соответствующего сообщества создавать новые приложения, которые поглощают все доступные ресурсы.

В этой главе мы сосредоточим внимание на производительности серверных систем. В настольных системах проблемы, характерные для серверов, обычно не возникают, поэтому ответ на вопрос о том, как повысить производительность настольной системы, обычно прост: “Провести модернизацию”. Пользователям такой ответ нравится, поскольку это означает, что у них появятся новые модные системы.

Одним из аспектов, отличающих UNIX и Linux от других основных операционных систем, является объем данных, которые они предоставляют о своих внутренних операциях. Детальная информация доступна буквально по каждому уровню системы, благода-

ря чему администраторы могут настраивать различные параметры именно так, как нужно. Если установить источник проблем с производительностью никак не удается, всегда можно просмотреть исходный код. По этим причинам UNIX и Linux часто считаются наиболее подходящими операционными системами для пользователей, которых особенно беспокоит тема производительности.

Даже при этих условиях настройка производительности не является простой задачей. Как пользователи, так и администраторы часто думают, что если бы они владели нужными "магическими" приемами, их системы работали бы в два раза быстрее. Одним из наиболее распространенных заблуждений является убеждение в том, что настроить производительность может помочь манипулирование переменными ядра, которые отвечают за управление системой страничного обмена и буферными пулами. В наши дни ядра основных дистрибутивов изначально настроены на достижение разумной (хотя и не оптимальной) производительности при различных уровнях загруженности.

Если попытаться оптимизировать систему по какому-то конкретному показателю производительности (например, по степени использования буферов), весьма вероятно, что поведение системы ухудшится в отношении других показателей и режимов работы. В этой главе будет рассказываться о том, как можно настроить производительность на уровне системы; возможность рассказать о настройке производительности на уровне приложений мы оставили другим. Как системному администратору, вам необходимо помнить о том, что разработчики — тоже люди. (Сколько раз вы говорили или думали, что проблема кроется в сети?) При таком уровне сложности современных приложений некоторые проблемы можно решить только посредством сотрудничества их разработчиков, системных администраторов, проектировщиков серверов, администраторов баз данных, администраторов памяти и архитекторов сети. В этой главе мы поможем вам определить, какие данные следует предъявить этим специалистам, чтобы они смогли решить проблемы производительности (если в действительности эти проблемы лежат в их области). Такой подход гораздо эффективнее, чем просто сказать: "Все выглядит прекрасно, это не моя проблема".

В любом случае, никогда не доверяйте полностью тому, что пишут в Интернете. Какие только аргументы не приводятся в дискуссиях о производительности систем! При этом сторонники всевозможных теорий не обладают, как правило, ни знаниями, ни дисциплиной, ни временем, необходимыми для проведения достоверных экспериментов. Широкая поддержка пользователей еще ничего не значит, ибо каждое глупое предложение очень часто сопровождается хором восторженных комментариев: "Я увеличил размер кеш-буфера в десять раз, как он предложил, и система заработала гораздо, гораздо быстрее!!!" Да уж, конечно!

Ниже перечислены некоторые правила, которыми следует пользоваться.

- Собирайте и анализируйте хронологические данные о работе системы. Если еще неделю назад производительность системы была нормальной, а сейчас все изменилось, то сравнение характеристик системы в двух состояниях позволит выявить источник проблемы. Всегда держите под рукой список оптимальных параметров производительности. А также первым делом просматривайте журнальные файлы, чтобы выяснить, не связана ли возникшая проблема с появлением какой-нибудь неполадки в оборудовании.
- В главе 21 рассказывалось о некоторых средствах трендового анализа, которые также подходят для мониторинга производительности. Утилита `sar`, которая будет описываться чуть позже в этой главе, тоже может использоваться в качестве средства трендового анализа.

- Всегда настраивайте систему так, чтобы потом иметь возможность сравнить результаты с предыдущими показателями производительности системы.
- Всегда проверяйте наличие плана отката на случай, если внесенные "чудодейственные" изменения окажутся вовсе не чудодейственными и лишь только ухудшат дело.
- Не перегружайте умышленно свои системы или свою сеть. Ядро создает для каждого процесса иллюзию бесконечности ресурсов. Но как только в дело вступают все 100% ресурсов системы, операционной системе приходится работать очень напряженно. На поддержку иллюзии расходуется ощутимая доля самих ресурсов. В результате выполнение процессов замедляется.
- Как в физике элементарных частиц, чем больше данных вы собираете с помощью утилит мониторинга системы, тем большее влияние вы оказываете на исследуемую систему. Для наблюдения за системой лучше всего полагаться на простые инструменты, которые работают в фоновом режиме (например, `sar` или `vmstat`). Если они обнаружат что-то существенное, то для более глубокого анализа можете воспользоваться и другими инструментами.

29.1. Способы повышения производительности

Ниже перечислены конкретные действия, которые можно выполнить, чтобы улучшить производительность.

- Убедиться в том, что память доступна в достаточном объеме. В следующем разделе вы увидите, что объем памяти очень сильно влияет на производительность. Устройства памяти сегодня стоят довольно недорого, так что обычно оснастить по максимуму любой компьютер, от которого требуется высокая производительность, — не проблема.
- Если UNIX- или Linux-система используется в качестве веб-сервера или сетевого сервера приложений, можно попробовать распределить трафик между несколькими компьютерами с помощью какого-нибудь коммерческого приложения для балансирования нагрузки, такого как Content Services Switch производства компании Cisco (`cisco.com`), ServerIron производства компании Brocade (`brocade.com`) или Alteon Web Switch производства компании Nortel (`nortelnetworks.com`).
- Эти устройства делают так, что несколько физических серверов выглядят для внешнего мира как один логический сервер. Они распределяют нагрузку согласно одному из нескольких выбираемых пользователем алгоритмов типа "минимальное время отклика" или "циклический алгоритм обслуживания".
- Тщательно проверить конфигурацию системы и отдельных приложений. Многие приложения могут резко увеличить производительность, если их правильно настроить (распределить данные между дисками, не осуществлять динамический поиск имен в DNS, запустить дополнительные экземпляры сервера и т.д.).
- Устранить проблемы, связанные с использованием ресурсов. Проблемы могут создаваться как "реальными заданиями" (одновременный запуск слишком большого количества серверов, неэффективные методы программирования, выполнение пакетов заданий с завышенным приоритетом, запуск ресурсоемких программ в неподходящее время), так и самой системой (ненужные демоны).
- Ликвидировать по возможности зависимость ресурсов памяти от механических операций. В настоящее время широко доступны полупроводниковые диски (Solid

state disk drives — SSDs), которые могут обеспечить быстрый рост производительности, поскольку они не требуют физического движения диска для считывания битов информации. SSD-диски легко устанавливаются вместо "старых добрых" дисковых накопителей[1].

- Организовать жесткие диски и файловые системы так, чтобы нагрузка на них была равномерной, и тем самым максимально повысить пропускную способность каналов ввода-вывода. При наличии специфических приложений, таких как базы данных, можно попробовать использовать какую-нибудь модную многодисковую технологию вроде RAID для оптимизации процессов обмена данными. За рекомендациями следует обращаться к производителю базы данных. Для систем Linux нужно удостовериться в том, что для диска выбран самый подходящий из доступных в Linux планировщик ввода-вывода.
- Обратить внимание на то, что разные приложения и СУБД ведут себя по-разному при размещении на нескольких дисках. Технология RAID поставляется в нескольких вариантах, поэтому следует потратить время и выяснить, какой из них лучше всего подходит для данного конкретного случая (если вообще подходит).
- Провести мониторинг сети, чтобы убедиться в том, что она не перегружена трафиком и что количество ошибок в ней минимально. Очень много такой полезной информации о сети можно собрать с помощью команды **`netstat`**, которая описывалась в разделе 21.5. Также можно еще раз перечитать главу 21.
- Определить ситуации, в которых система совершенно не удовлетворяет предъявляемым к ней требованиям. Нельзя настраивать производительность без учета таких ситуаций.

Эти меры перечислены в порядке убывания эффективности. Добавление памяти и распределение трафика между несколькими серверами может значительно повысить производительность. Степень эффективности остальных мер варьируется от значительной до нулевой.

Анализ и оптимизация структур данных и алгоритмов программ почти всегда ведет к существенному повышению производительности. Однако эти вопросы обычно находятся вне компетенции системного администратора.

29.2. Факторы, влияющие на производительность

Производительность системы во многом определяется базовыми характеристиками системных ресурсов и эффективностью их распределения и совместного использования. Определение термина "ресурс" весьма расплывчато. Оно может подразумевать даже такие компоненты, как кеш содержимого регистров центрального процессора и элементы таблицы адресов контроллера памяти. Однако в первом приближении серьезное влияние на производительность оказывают только четыре фактора:

- время использования центрального процессора;
- память;

[1] У современных SSD-дисков есть два основных недостатка. Во-первых, они на порядок дороже (в расчете на гигабайт), чем традиционные жесткие диски. Во-вторых, их можно перезаписывать только ограниченное число раз. Их перезаписывающая способность достаточно высока и вполне приемлема для настольных компьютеров (десятки тысяч записей на блок), но может оказаться потенциальным камнем преткновения для сервера с большой рабочей нагрузкой. Подробнее о SSD-дисках см. в разделе 8.2.

- пропускная способность дисковой подсистемы ввода-вывода;
- пропускная способность сетевой подсистемы ввода-вывода.

Если после выполнения активных процессов остались свободные ресурсы, можно считать, что производительность системы является удовлетворительной.

Когда ресурсов не хватает, процессы становятся в очередь. Процесс, не имеющий немедленного доступа к необходимым ресурсам, должен ждать, ничего при этом не делая. Время, затрачиваемое на ожидание ресурсов, — один из основных показателей ухудшения производительности.

Самый простой, с точки зрения учета, ресурс — использование центрального процессора (ЦП). В распоряжении системы всегда имеется примерно одна и та же часть его мощности. Теоретически эта величина составляет все 100% циклов ЦП, но "накладные расходы" и другие причины приводят к снижению этого показателя примерно до 95%. Для процесса, занимающего более 90% времени ЦП, ограничивающим фактором является быстродействие центрального процессора. Такой процесс потребляет большую часть вычислительных мощностей системы.

Многие считают, что быстродействие ЦП — основной фактор, влияющий на общую производительность системы. При наличии неограниченных объемов всех остальных ресурсов или для определенных типов приложений (например, программ численного моделирования) быстродействие ЦП действительно играет роль. Но в повседневной жизни этот показатель значит относительно мало.

Настоящим узким местом является канал обмена данными с жестким диском. Жесткие диски представляют собой механические системы, поэтому на поиск нужного блока, выборку его содержимого и активизацию ожидающего его процесса уходят десятки миллисекунд. Задержки такого порядка отодвигают на второй план все остальные факторы ухудшения производительности. Каждое обращение к диску вызывает приостановку активного процесса "стоимостью" несколько миллионов инструкций ЦП. Эту проблему можно решить, например, с помощью полупроводниковых дисковых накопителей, которые работают значительно быстрее, чем диски с движущимися деталями.

Благодаря использованию виртуальной памяти, скорость дискового обмена может быть непосредственно связана с объемом имеющейся памяти, если потребность в физической памяти превышает ее реальный объем. Ситуации, в которых физической памяти становится недостаточно, часто приводят к необходимости записывать содержимое ее страниц на диск, чтобы эти страницы можно было восстановить и использовать для другой цели. Это означает, что обращение к памяти по своей "стоимости" приближается к работе с диском. Избегайте таких ловушек, если характеристики производительности имеют для вас большое значение; побеспокойтесь о том, чтобы каждая система имела достаточно физической памяти.

Пропускная способность сети подвержена примерно тем же ограничениям, что и скорость дискового обмена. Ее ухудшение связано со всевозможными задержками. Но возникающие проблемы охватывают целые сообщества пользователей, а не отдельные компьютеры. Кроме того, сети восприимчивы к проблемам аппаратного обеспечения и перегрузкам серверов.

29.3. Как анализировать проблемы производительности

В сложной системе нелегко выделить именно проблемы производительности. Системные администраторы часто получают частные (несистематические) отчеты о пробле-

мах, которые содержат эмоциональные описания конкретных случаев или сложных ситуаций (например, “веб-сервер застопорился из-за всех этих проклятых AJAX-вызовов...”). Вы, конечно, должны обратить внимание на эту информацию, но не воспринимайте ее как достоверную и надежную; проводите собственное исследование.

Строгая и прозрачная научная методика поможет вам сделать правильные выводы, которым ваши сотрудники и вы сами можете доверять. Научный подход позволит другим оценить ваши результаты, повысит доверие к вашей работе и увеличит вероятность того, что предлагаемые вами изменения смогут реально решить проблему.

Применение научного подхода не означает, что вы должны собирать все релевантные данные сами. Весьма полезной бывает и “внешняя” информация. Не стоит тратить часы на изучение вопросов, ответы на которые можно легко почерпнуть из разделов FAQ (Frequently Asked Questions — часто задаваемые вопросы).

Мы предлагаем вам выполнить следующие пять действий.

Шаг 1. Сформулируйте вопрос.

Поставьте конкретный вопрос в определенной функциональной области или сформулируйте предварительное заключение или рекомендацию, которую вы обдумали. Будьте точны, говоря о технологиях, компонентах и альтернативах, которые вы предлагаете рассмотреть, и результатах, которые могут быть достигнуты.

Шаг 2. Соберите и классифицируйте факты.

Проведите систематический поиск в документации, базах знаний, известных вам изданиях, блогах, технических описаниях, форумах и прочих информационных ресурсах с целью выявления внешних фактов, имеющих отношение к вашему вопросу. В собственных системах зафиксируйте телеметрические данные и (по возможности или необходимости) оснастите инструментальными средствами конкретные интересующие вас области в системе или отдельных приложениях.

Шаг 3. Критически оцените данные.

Просмотрите каждый источник данных на предмет релевантности и критически оцените его с точки зрения достоверности. Выделите ключевые данные и обратите внимание на качество источников информации.

Шаг 4. Подытожьте данные вербально и графически.

Представьте сведения, взятые из нескольких источников, в словесной (вербальной) и по возможности графической форме. Данные, кажущиеся сомнительными при выражении в числовой форме, могут оказаться убедительными в виде диаграммы.

Шаг 5. Сформулируйте заключение.

Представьте свои выводы в лаконичной форме (как ответ на поставленный вами же вопрос). Поставьте оценку, чтобы обозначить уровень силы или слабости доказательств, которые поддерживают ваши заключения.

29.4. Проверка производительности системы

Хватит обобщать — пора рассмотреть некоторые конкретные инструменты и “зоны интереса”. Прежде чем переходить к измерениям, необходимо знать, что именно вы хотите получить.

Инвентаризуйте свое оборудование

Начните с оценки своего оборудования, особенно ЦП и ресурсов памяти. Инвентаризация поможет вам интерпретировать данные, представленные с помощью других

инструментов, а также построить реалистичные ожидания касательно верхних границ производительности.

В системах Linux именно в файловой системе **/proc** стоит искать ответ на вопрос, какое, с точки зрения вашей операционной системы, вы используете оборудование (более детальную информацию об аппаратуре можно найти в каталоге **/sys**; см. раздел 13.8). Некоторые ключевые файлы перечислены в табл. 29.1. Общую информацию о файловой системе **/proc** можно найти в разделе 13.3.

Таблица 29.1. Источники информации об оборудовании в системах Linux

Файл	Содержимое
/proc/cpuinfo	Тип и описание ЦП
/proc/meminfo	Размер памяти и коэффициент загрузки
/proc/diskstats	Дисковые устройства и статистика их использования

Четыре строки в файле **/proc/cpuinfo** помогут вам идентифицировать ЦП в системе: `vendor_id`, `cpu family`, `model`, и `model name`. Некоторые из этих значений непонятны, поэтому лучше всего узнать о них в интерактивной справочной системе.

Точная информация, содержащаяся в файле **/proc/cpuinfo**, зависит от системы и процессора. Рассмотрим типичный пример содержимого этого файла.

```
suse$  cat /proc/cpuinfo
processor     : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 15
model name    : Intel(R) Xeon(R) CPUE5310@ 1.60GHz
stepping      : 11
cpu MHz       : 1600.003
cache size    : 4096 KB
physical id   : 0
cpu cores     : 2
siblings      : 2
…
```

Этот файл содержит по одной записи для каждого процессорного ядра, видимого операционной системой. Конкретные данные незначительно варьируются в зависимости от версии ядра. Значение поля `processor` уникально идентифицирует ядро. Значения поля `physical id` являются уникальными для каждого физического сокета на печатной плате, а значения `id values` уникальны для ядра в физическом сокете. Ядра, которые поддерживают гиперпотоковый режим работы (дублирование контекстов ЦП без дублирования других характеристик обработки), идентифицируются значением `ht` в поле `flags`. Если в системе действительно реализован гиперпотоковый режим работы, поле `siblings` для каждого ядра показывает, сколько контекстов доступно на данном ядре.

Существует еще одна команда, которая позволяет получить информацию об аппаратных характеристиках вашего компьютера. Речь идет о команде **dmidecode**. Она выводит содержимое системной таблицы DMI (Desktop Management Interface — интерфейс управления настольными системами), также известной под именем SMBIOS. Самой полезной опцией этой команды является **-t** *тип* (допустимые типы представлены в табл. 29.2).

Таблица 29.2. Значения типов для команды `dmidecode -t`

Значение	Описание
1	Система
2	Материнская плата
3	Корпус
4	Процессор
7	Кеш-память
8	Разъемы портов
9	Системные разъемы
11	OEM-строки
12	Опции системной конфигурации
13	Язык BIOS
16	Массив физической памяти
17	Устройство памяти
19	Отображаемый адрес массива памяти
32	Загрузка системы
38	IPMI-устройство

Следующий пример демонстрирует получение типичной информации.

```
suse$  sudo dmidecode -t 4
# dmidecode 2.7
SMBIOS 2.2 present.

Handle 0x0004, DMI type 4, 32 bytes.
Processor Information
   Socket Designation: PGA 370
   Type: Central Processor
   Family: Celeron
   Manufacturer: GenuineIntel
   ID: 65 06 00 00 FF F9 83 01
   Signature: Type 0, Family 6, Model 6, Stepping 5
```

Биты информации о сетевой конфигурации разбросаны “по всей системе”. Лучшим источником IP- и MAC-информации для каждого сконфигурированного интерфейса служит команда `ifconfig -a`.

В системах Solaris лучшим источником информации о ЦП и ресурсах памяти являются команды `psrinfo -v` и `prtconf` соответственно. Вот пример результата выполнения этих команд.

```
solaris$ psrinfo -v
Status of virtual processor 0 as of: 01/31/2010 21:22:00
   on-line since 07/13/2009 15:55:48.
   The sparcv9 processor operates at 1200 MHz,
      and has a sparcv9 floating point processor.
Status of virtual processor 1 as of: 01/31/2010 21:22:00
   on-line since 07/13/2009 15:55:49.
   The sparcv9 processor operates at 1200 MHz,
      and has a sparcv9 floating point processor.
```

```
solaris$ prtconf
System Configuration:  Sun Microsystems  sun4v
Memory size: 32640 Megabytes
System Peripherals (Software Nodes):

SUNW,Sun-Fire-T200
…
```

В системах HP-UX информацию о конфигурации аппаратных средств компьютера можно получить с помощью одной-единственной команды `machinfo`. Вот типичный пример ее выполнения.

```
hp-ux$  sudo machinfo
CPU info:
   1 Intel(R) Itanium 2 processor (1.5 GHz, 6 MB)
      400 MT/s bus, CPU version B1

Memory: 4084 MB (3.99 GB)

Firmware info:
   Firmware revision:      02.21
   FP SWA driver revision: 1.18
   BMC firmware revision:  1.50

Platform info:
   Model:       "ia64 hp server rx2600"
OS info:
   Nodename:    hpux11
   Release:    HP-UX B.11.31
Machine:       ia64
```

В системах AIX для получения информации о ЦП и памяти придется выполнить несколько действий. Сначала с помощью команды **`lscfg`** найдите имена инсталлированных процессоров.

```
aix$  lscfg | grep Processor
+ proc0                    Processor
+ proc2                    Processor
```

Затем для получения описания каждого процессора можно использовать команду `lsattr`.

```
aix$  lsattr -E -l proc0
frequency      1898100000       Processor    Speed          False
smt_enabled    true             Processor    SMT enabled    False
smt_threads    2                Processor    SMT threads    False
state          enable           Processor    state          False
type           PowerPC_POWER5   Processor    type           False
```

С помощью команды `lsattr` можно также узнать объем физической памяти в системе.

```
aix$  lsattr -E -l sys0 -a realmem
realmem       4014080       Amount of usable physical memory in Kbytes
```

Сбор данных о производительности

Программы анализа производительности в большинстве своем сообщают о том, что происходит в системе в данный момент времени. Но следует учесть, что структура и характер нагрузки меняются в течение дня. Поэтому до принятия мер обязательно проведите всесторонний анализ данных, касающихся производительности. Истинная производительность выясняется только после длительного (месяц или даже больше) наблюдения за системой. Особенно важно собирать данные в периоды пиковой загруженности системы. Ограничения на использование ресурсов и ошибки в конфигурации системы часто выявляются только в таких условиях.

Анализ использования центрального процессора

Обычно собирают три вида данных о центральном процессоре: общий показатель использования, показатели средней загруженности и потребление ресурсов отдельными процессами. Первая характеристика определяет, является ли быстродействие самого процессора узким местом. Показатели средней загруженности дают возможность оценить общую производительность системы. Данные, касающиеся отдельных процессов, позволяют выявить наиболее ресурсоемкие процессы.

Сводную информацию выдает команда `vmstat`. Она принимает два аргумента: время (в секундах), в течение которого необходимо наблюдать за системой для получения каждой строки выходной информации, и необходимое количество отчетов. Если не указать число отчетов, команда будет выполняться до тех пор, пока пользователь не нажмет комбинацию клавиш <Ctrl+C>.

В первой строке отображаемых данных сообщаются средние значения, измеренные с момента запуска системы. В последующих строках выдаются результаты по каждому очередному замеру, стандартная продолжительность которого составляет пять секунд.

```
$ vmstat 5 5
procs  -----------memory---------  -swap- --io--  -system--  ----cpu----
r b swpd     free  buff     cache  si so   bi bo    in   cs  us sy id wa
1 0  820  2606356  428776  487092   0  0 4741 65  1063 4857  25  1 73  0
1 0  820  2570324  428812  510196   0  0 4613 11  1054 4732  25  1 74  0
1 0  820  2539028  428852  535636   0  0 5099 13  1057 5219  90  1  9  0
1 0  820  2472340  428920  581588   0  0 4536 10  1056 4686  87  3 10  0
3 0  820  2440276  428960  605728   0  0 4818 21  1060 4943  20  3 77  0
```

Несмотря на то что содержимое этих колонок может не совпадать в разных системах, статистические данные показателей использования ЦП практически не различаются среди платформ. Пользовательское время, системное время (время ядра), время простоя и время ожидания для каналов ввода-вывода (I/O) показаны в колонках `us`, `sy`, `id` и `wa` соответственно. Большие числа в колонке `us` обычно означают вычисления, а в колонке `sy` они свидетельствуют о том, что процессы осуществляют очень много системных вызовов или выполняют операции ввода-вывода.

Выработанное нами за многие годы эмпирическое правило для серверов общего назначения, справедливое для большинства систем, гласит следующее: система должна тратить примерно 50% рабочего времени на обслуживание пользовательских запросов и столько же на системные запросы. Общее время простоя не должно быть нулевым. Если сервер выделяется под одно единственное, но интенсивно использующее ЦП приложение, большая часть времени должна тратиться на обработку пользовательских запросов.

В колонке cs показано число переключений контекста за данный период, т.е. сколько раз ядро переключало процессы. В колонке in отображается число прерываний, генерируемых аппаратными устройствами или компонентами ядра. Слишком большие значения в этих колонках свидетельствуют о неправильно работающем аппаратном устройстве. Остальные колонки важны для анализа использования памяти и жесткого диска, о чем будет рассказываться позже в этой главе.

Длительные измерения показателей, характеризующих работу центрального процессора, позволяют определить, достаточно ли его ресурсов для нормальной работы системы. Если процессор регулярно часть времени простаивает, значит, есть запас по тактовой частоте. В этом случае увеличение быстродействия процессора ненамного улучшит общую производительность системы, хотя может и ускорить выполнение отдельных операций.

Как видно из показанного примера, центральный процессор постоянно переключается из режима интенсивного использования в режим полного бездействия и обратно. Поэтому необходимо наблюдать за показателями, соответствующими обоим режимам, и выводить среднее за определенный период времени. Чем меньше интервал наблюдения, тем ниже достоверность оценок.

В мультипроцессорных системах команды вроде **vmstat** сообщают средние значения по всем процессорам. Но существует и команда **mpstat**, которая выдает отчет по каждому процессору. В системах Linux, Solaris и AIX с помощью флага -P можно выбрать один конкретный процессор. Этой командой удобно пользоваться при отладке программ, поддерживающих симметричную многопроцессорную обработку. Интересно также узнать, насколько система эффективно (или неэффективно) работает с несколькими процессорами.

Рассмотрим пример отображения статуса каждого из четырех процессоров.

```
linux$ mpstat -P ALL
08:13:38 PM CPU %user %nice %sys %iowait %irq %soft %idle intr/s
08:13:38 PM   0  1.02  0.00 0.49    1.29 0.04  0.38 96.79 473.93
08:13:38 PM   1  0.28  0.00 0.22    0.71 0.00  0.01 98.76 232.86
08:13:38 PM   2  0.42  0.00 0.36    1.32 0.00  0.05 97.84 293.85
08:13:38 PM   3  0.38  0.00 0.30    0.94 0.01  0.05 98.32 295.02
```

Центральный процессор однопользовательской рабочей станции обычно простаивает бо́льшую часть времени. Когда запрашивается прокрутка содержимого окна или обновляется веб-страница, ЦП справляется с этой операцией за считанные секунды. В такой ситуации долгосрочный средний показатель использования ЦП практически лишен смысла.

Еще одним статистическим показателем, полезным для оценки интенсивности использования системы, является показатель “средняя загруженность”, который представляет собой среднее количество выполняемых процессов. Он дает достаточное представление о том, сколько процессов требуют свою долю процессорного времени. Узнать его значение можно с помощью команды uptime.

```
$ uptime
11:10am  up 34 days, 18:42, 5 users, load average: 0.95, 0.38, 0.31
```

Выдаются три значения, которые соответствуют средней загруженности за пять, десять и пятнадцать минут. Как правило, чем выше средняя загруженность, тем важнее общая производительность системы. Если выполняется всего лишь один процесс, то он обычно ограничен одним ресурсом (центральным процессором или пропускной способ-

ностью дискового канала ввода-вывода). Пиковый спрос на этот ресурс и становится определяющим фактором производительности.

Когда в системе одновременно работает несколько процессов, нагрузка распределяется более равномерно. Если все работающие процессы потребляют ресурсы ЦП, диска и памяти, то зависимость производительности системы от ограниченных возможностей какого-то одного ресурса снижается. В такой ситуации гораздо важнее следить за средними показателями загруженности, в частности за общим временем использования центрального процессора.

📖 О приоритетах рассказывалось в конце раздела 5.1.

Обычно в однопроцессорных системах значение 3 показателя средней загруженности свидетельствует о сильной загруженности системы, а значение 8 и выше — считается критическим. Такие значения являются сигналом о том, что пора начать поиск путей искусственного перераспределения нагрузки, например назначить процессам приоритеты с помощью команды `nice`.

Показатель средней загруженности отлично подходит для повседневного контроля системы. Если известен показатель работающей системы и он находится в диапазоне, характерном для перегрузки, значит, следует искать внешний источник проблемы (это может быть, к примеру, сеть). В противном случае нужно проверить процессы самой системы.

Еще один способ контроля над использованием ЦП — посмотреть, какую часть его времени отнимает каждый процесс. Для этого служит команда **`ps`** с аргументами (`-aux` — в системах Linux и AIX; `-elf` — в системах HP-UX и Solaris). Зачастую в интенсивно эксплуатируемой системе минимум 70% времени ЦП отнимается одним-двумя процессами. Запуск таких процессов в другое время суток или снижение их приоритетов позволит высвободить ЦП для выполнения других процессов.

📖 Подробнее об утилите **`top`** рассказывалось в разделе 5.8.

Прекрасной альтернативой команде **`ps`** является утилита **`top`**. Она выдает примерно ту же информацию, что и `ps`, но в “живом”, постоянно обновляемом формате, позволяющем наблюдать за изменением состояния системы во времени[2]. В системе AIX можно воспользоваться даже более эффективной командой **`topas`**.

В виртуализированных системах **`ps`**, **`top`** и другие команды, которые отображают данные о показателях использования ЦП, могут вводить в заблуждение. Виртуальная машина, которая не использует все виртуальные циклы своего центрального процессора, позволяет другим виртуальным машинам использовать (заимствовать) эти циклы. Любое измерение, которое имеет отношение к операционной системе (например, количество “тиков” системных часов в секунду), должно быть вами тщательно проанализировано, чтобы вы до конца понимали результаты своих измерений. О различных технологиях виртуализации можно подробнее см. в главе 24.

Управление памятью в системе

В UNIX- и Linux-системах память организована в виде модулей, называемых *страницами*, размер которых составляет 4 Кбайт или больше. Когда процессы запрашивают у операционной системы память, ядро выделяет им виртуальные страницы. Каждой такой странице соответствует настоящее физическое хранилище, такое как ОЗУ или “резерв-

[2] Утилита *`top`* потребляет довольно много ресурсов, поэтому пользуйтесь ею в разумных пределах.

ное ЗУ" на жестком диске. (Резервное ЗУ обычно представляет собой просто пространство в области подкачки, но для страниц, которые содержат текст исполняемой программы, резервное ЗУ — это самый настоящий исполняемый файл. Аналогично резервное ЗУ для некоторых файлов данных может представлять собой сами файлы.) Информация о связях между виртуальными и реальными страницами хранится в таблице страниц.

Ядро способно эффективно обслуживать запросы процессов, выделяя для них столько памяти, сколько им нужно. Это реализуется дополнением реального ОЗУ областью подкачки. Поскольку процессы ожидают, что их виртуальные страницы будут отображаться в реальной памяти, ядру постоянно приходится перемещать страницы между ОЗУ и областью подкачки. Такие операции называются *страничным обменом* (paging)[3].

Ядро старается управлять памятью так, чтобы страницы, к которым недавно обращались, хранились в физической памяти, а менее активные страницы выгружались на диск. Этот алгоритм известен под названием LRU (least recently used — замещение наименее используемых элементов), поскольку те страницы, к которым долго никто не обращался, перемещаются на диск.

Впрочем, отслеживать все обращения к страницам было бы слишком накладно для ядра, поэтому оно использует страничный кеш-буфер, анализ содержимого которого и позволяет принять решение о том, какие именно страницы оставить в памяти. Точный алгоритм этого механизма зависит от конкретной системы, но везде действует примерно одинаковый принцип. Такой вариант гораздо проще в реализации, чем LRU-система, а результаты дает почти такие же.

В случае нехватки памяти ядро пытается определить, какие страницы из "неактивного" списка давно не использовались. Если при последнем обращении такие страницы модифицировались процессом, ядро считает их "грязными"; они обязательно должны быть выгружены на диск, прежде чем память будет повторно использована. Все остальные страницы считаются "чистыми" и могут быть переданы другому процессу.

Когда выполняется обращение к странице из "неактивного" списка, ядро возвращает ссылку на нее в таблицу страниц, обнуляет ее "возраст" и переводит страницу из "неактивного" списка в "активный". Страницы, выгруженные на диск, должны быть прочитаны с диска, прежде чем их можно будет активизировать. Когда процесс обращается к неактивной странице, находящейся в памяти, происходит *сбой* программной страницы (это так называемая "мягкая ошибка"). В случае обращения к нерезидентной (выгруженной) странице имеет место *отказ* страницы диска (а это уже "жесткая ошибка"). Другими словами, при возникновении ошибки второго типа (в отличие от первого) требуется прочитать страницу с диска.

Ядро старается прогнозировать потребности системы в памяти, поэтому операции выделения и выгрузки страниц не обязательно взаимосвязаны. Цель системы — обеспечить наличие достаточного объема свободной памяти, чтобы процессам не приходилось ждать выгрузки страниц всякий раз, когда им нужна свободная память. Если в периоды сильной загруженности системы страничный обмен резко возрастает, имеет смысл купить дополнительную память.

Система Linux по-прежнему быстро эволюционирует, и ее система виртуальной памяти продолжает развиваться, причем несколько неровно и даже неуклюже. Можно настроить параметр "подкачки" (**`/proc/sys/vm/swappiness`**) и тем самым дать ядру подсказку о том, насколько быстро оно должно делать

[3] Когда-то имела место иная операция, именуемая *подкачкой* (или *свопингом*), посредством которой все страницы некоторого процесса одновременно переписывались на диск. Теперь же во всех случаях обязательно используется страничный обмен (paging).

страницы пригодными для возврата из процесса в случае нехватки памяти. По умолчанию для этого параметра устанавливается значение 60. Если для него установить значение 0, ядро будет забирать страницы, которые были назначены процессу, только тогда, когда испробует все остальные возможности. Если для него установить значение 60 или выше (максимальным значением является 100), ядро, скорее всего, будет забирать страницы из процесса. Если возникает желание изменить этот параметр, значит, возможно, пришла пора увеличить ОЗУ.

Когда ядру не хватает как физической памяти, так и области подкачки, это означает, что виртуальная память исчерпана. В данной ситуации включается режим "принудительного уничтожения". Чтобы освободить память, системе приходится уничтожать целые процессы. И хотя выбираются наименее важные для системы процессы, все равно это очень неприятная процедура, ведь значительная часть ресурсов системы тратится не на полезную работу, а на управление памятью.

Анализ использования памяти

Интенсивность операций с памятью количественно представляется двумя показателями: общим объемом задействованной виртуальной памяти и страничного обмена. Первый показатель характеризует общую потребность в памяти, а второй указывает на то, какая доля этой памяти активно используется. Задача администратора заключается в снижении интенсивности использования или увеличении объема памяти до тех пор, пока не будет достигнут приемлемый уровень страничного обмена. Страничный обмен — процесс неизбежный, поэтому не пытайтесь полностью избавиться от него.

У вас есть возможность определить размер текущей области подкачки. Для этого в Linux выполните команду **swapon -s**, в Solaris и AIX — команду **swap -l**, а в HP-UX — команду **swapinfo**.

```
linux$  swapon -s
Filename   Type       Size     Used  Priority
/dev/hdb1  partition  4096532  0     -1
/dev/hda2  partition  4096564  0     -2
solaris$  swap -l
swapfile           dev   swapl  blocks   free
/dev/dsk/c0t0d0s1  32,1     16  164400 162960
hp-ux$  swapinfo
          Kb    Kb       Kb   PCT START/       Kb
TYPE   AVAIL  USED     FREE  USED  LIMIT  RESERVE  PRI           NAME
dev  8388608     0  8388608    0%      0        -    1  /dev/vg00/lvol
```

При выполнении команд **swapinfo** и **swapon** значения выводятся в килобайтах, а команда **swap -l** использует 512-байтовые дисковые блоки. Значения размеров, выводимые этими командами, не включают содержимое памяти ядра, поэтому вам придется вычислить общий объем виртуальной памяти самостоятельно.

```
VM = ОЗУ + используемый_объем_области_подкачки
```

В системах UNIX вывод статистических показателей страничного обмена, полученных с помощью команды `vmstat`, подобен результату выполнения команды в системе Solaris.

```
solaris$ vmstat 5 5
procs       memory            page            disk          faults
```

```
r b w    swap   free  re  mf pi po fr de sr  s0 s6 s4 --  in  sy  cs
0 0 0  338216  10384   0   3  1  0  0  0  0   0  0  0  0  132 101 58
0 0 0  341784  11064   0  26  1  1  1  0  0   0  0  1  0  150 215 100
0 0 0  351752  12968   1  69  0  9  9  0  0   0  0  2  0  173 358 156
0 0 0  360240  14520   0  30  6  0  0  0  0   0  0  1  0  138 176 71
1 0 0  366648  15712   0  73  0  8  4  0  0   0  0 36  0  390 474 237
```

Из этого примера удалена информация об использовании центрального процессора. Под заголовком `procs` показано количество процессов, готовых к немедленному выполнению, заблокированных в ожидании ввода-вывода, а также готовых к выполнению, но перекачанных на диск. Если значение в колонке `w` когда-нибудь станет отличным от нуля, это будет означать, что объем системной памяти не соответствует текущей нагрузке.

Колонки, объединенные под общим заголовком `page`, содержат данные о выполнении страничного обмена. Во всех этих колонках представлены средние значения: количество в секунду (табл. 29.3).

В колонке `swap` выдается объем доступной виртуальной памяти в килобайтах. В колонке `free` отображается объем (в килобайтах) списка неиспользуемых страниц системы. Если приведенные в ней числа не достигают 3% от общего объема системной памяти, это говорит о наличии проблем.

Таблица 29.3. Описание статистических показателей, выводимых командой `vmstat`

Колонка	Описание показателя
`re`	Количество восстановленных страниц, т.е. возвращенных из списка неиспользуемых
`mf`	Количество незначительных ошибок (связанных с небольшим числом страниц)
`pi`	Объем подкачанных данных в килобайтах
`po`	Объем выгруженных данных в килобайтах
`fr`	Объем списка неиспользуемых страниц в килобайтах
`de`	Объем “ожидаемого краткосрочного дефицита памяти” в килобайтах
`sr`	Количество страниц, обработанных по алгоритму часов

Содержимое колонки `de` — самый надежный показатель возникновения серьезных проблем с памятью. Если находящееся в ней число зачастую “зашкаливает” за 100, это значит, что компьютеру нужно больше памяти. К сожалению, во многих версиях команды `vmstat` это значение не выводится.

В системах Linux статистические показатели, получаемые с помощью команды `vmstat`, могут иметь следующий вид.

```
linux$ vmstat 5 5
procs -------memory--------- -swap- ---io--  -system-  -----cpu------
 r b  swpd  free  buff  cache  si so   bi bo   in  cs  us sy id wa st
 5 0     0 66488 40328 597972   0  0  252 45  1042 278   3  4 93  1  0
 0 0     0 66364 40336 597972   0  0    0 37  1009 264   0  1 98  0  0
 0 0     0 66364 40344 597972   0  0    0  5  1011 252   1  1 98  0  0
 0 0     0 66364 40352 597972   0  0    0  3  1020 311   1  1 98  0  0
 0 0     0 66364 40360 597972   0  0    0 21  1067 507   1  3 96  0  0
```

Как и в результате выполнения команды **`vmstat`** при использовании системы UNIX, количество процессов, готовых к немедленному выполнению и заблокированных в ожидании ввода-вывода, выводится под заголовком **`procs`**. Статистические показатели

страничного обмена ограничены двумя колонками: si и so, которые представляют количество подкачанных и выгруженных страниц соответственно.

Кажущиеся несоответствия между колонками, большей частью, иллюзорны. В одних колонках указывается число страниц, в других — объем в килобайтах. Все значения являются округленными средними величинами. Более того, одни из них — средние скалярных величин, а другие — средние изменений.

С помощью полей si и so можно оценить интенсивность страничного обмена в системе. Операция загрузки (поле si) не обязательно означает, что из области подкачки восстанавливается страница. Это может быть исполняемый код, постранично загружаемый из файловой системы, или копия страницы, создаваемая в режиме дублирования при записи. Оба случая вполне типичны и не обязательно означают нехватку памяти. С другой стороны, операция выгрузки (поле so) всегда означает принудительное изъятие данных ядром.

Система, в которой непрерывно происходят операции выгрузки, скорее всего, выиграет от добавления памяти. Если же это случается лишь время от времени и не вызывает жалоб со стороны пользователей, то можно не беспокоиться. В остальных случаях дальнейший анализ будет зависеть от того, что нужно сделать, — оптимизировать систему для интерактивного режима (например, как рабочую станцию) или сконфигурировать ее для одновременной работы пользователей (например, как сервер приложений).

При использовании обычного жесткого диска можно считать, что каждые 100 операций выгрузки создают задержку примерно в одну секунду[4]. Соответственно, если для прокрутки окна требуется 150 операций выгрузки, придется ждать около полутора секунды. Исследователи пользовательских интерфейсов утверждают, что среднестатистический пользователь считает систему “медленной”, когда время ее реакции превышает семь десятых секунды.

Анализ операций обмена с диском

Производительность жесткого диска можно контролировать с помощью команды iostat. Как и vmstat, эта команда поддерживает дополнительные аргументы, задающие интервал времени в секундах между моментами выдачи статистических данных за истекший период и число повторений. Команда iostat выдает также сведения об использовании ЦП. Приведем небольшой пример для системы Solaris.

```
solaris$  iostat 5 5
   tty        sd0           sd1           nfs1           cpu
tin tout  kps tps serv  kps tps serv  kps tps serv  us sy wt  id
  0    1    5   1   18   14   2   20    0   0    0   0  0  0  99
  0   39    0   0    0    2   0   14    0   0    0   0  0  0 100
  2   26    3   0   13    8   1   21    0   0    0   0  0  0 100
  3  119    0   0    0   19   2   13    0   0    0   0  1  1  98
  1   16    5   1   19    0   0    0    0   0    0   0  0  0 100
```

Колонки разделены по темам (в данном случае их пять: tty, sd0, sdl, nfsl и cpu). В разных системах данные, выводимые командой iostat, выглядят по-разному.

В разделе tty содержатся данные о терминалах и псевдотерминалах. В общем-то, это неинтересная информация, хотя она и может оказаться полезной для оценки пропускной способности модема. В колонках tin и tout дается среднее суммарное число символов, введенных и выведенных всеми терминалами системы за секунду.

[4] Мы считаем, что половину операций страничного обмена составляет выгрузка.

Информация о каждом жестком диске размещается в колонках kps, tps и serv: объем данных, пересланных за секунду (в килобайтах); общее количество пересылок за секунду; и среднее время позиционирования головки в миллисекундах. Одно обращение к диску может затронуть сразу несколько его секторов, поэтому соотношение между числами, приведенными в колонках kps и tps, говорит о структуре пересылок: то ли это несколько крупных пересылок, то ли множество мелких. Крупные пересылки более эффективны. Механизм вычисления времени позиционирования, похоже, работает только на некоторых дисковых накопителях и иногда дает странные результаты (к данному примеру это не относится).

Результат выполнения команды **iostat** в системах Linux, HP-UX и AIX может выглядеть примерно так.

```
aix$ iostat
…
Device:  tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
hdisk0  0.54        0.59        2.39    304483   1228123
hdisk1  0.34        0.27        0.42    140912    216218
hdisk2  0.01        0.02        0.05      5794     15320
hdisk3  0.00        0.00        0.00         0         0
```

Для каждого жесткого диска сообщается число операций ввода-вывода в секунду (tps), количество операций блочного чтения и блочной записи в секунду (Blk_read/s и Blk_write/s), общий объем прочитанных (Blk_read) и записанных (Blk_wrtn) блоков.

Размер дискового блока обычно равен 1 Кбайт (KiB), поэтому можно легко определить реальную скорость обмена данными с диском в килобайтах в секунду. В то же время понятие операции ввода-вывода несколько размыто. Один запрос на пересылку данных может включать в себя несколько запросов ввода-вывода к различным секторам.

Время, затрачиваемое на поиск блока, — основной фактор, влияющий на производительность жесткого диска. В первом приближении скорость вращения диска и быстродействие шины, к которой он подключен, не имеют особого значения. Современные диски могут пересылать сотни мегабайтов данных в секунду, если эти данные считываются из смежных секторов. Вместе с тем количество операций поиска составляет от 100 до 300 в секунду. Если после каждой операции поиска будет читаться один-единственный сектор, легко подсчитать, что в таком режиме задействуется менее 5% максимальной пропускной способности канала обмена данными с диском.

Затраты времени на поиск блока возрастают, если головке приходится перемещаться на большие расстояния. Когда есть диск с файловой системой, расположенной в нескольких разделах, и файлы считываются из каждого раздела в случайной последовательности, то для перехода из одного раздела в другой головка вынуждена преодолевать очень большой путь. С другой стороны, файлы в одном разделе расположены относительно близко друг к другу. Разбивая новый диск на разделы, необходимо принять во внимание факторы, влияющие на производительность, и постараться поместить файлы, обращение к которым осуществляется одновременно, в одну файловую систему.

Для достижения максимальной производительности нужно помещать файловые системы, которые используются одновременно, на разные диски. Хотя тип шины и драйверы устройств влияют на степень эффективности, большинство компьютеров может работать с несколькими дисками параллельно, что значительно повышает пропускную способность дисковой подсистемы. Например, данные и журнальные файлы веб-сервера имеет смысл размещать на разных дисках.

Особенно важно распределить область подкачки между несколькими дисками, если это возможно, поскольку страничный обмен обычно замедляет работу системы в целом. Многие системы позволяют создавать как выделенные разделы подкачки, так и файлы подкачки, записываемые в обычную файловую систему.

Кроме того, многие системы позволяют создавать "резидентные" файловые системы. Фактически это то же самое, что и виртуальный диск ПК. Специальный драйвер выдает себя за драйвер диска, хотя на самом деле записывает данные в память. Во многих организациях виртуальные диски применяются для хранения файловой системы `/tmp` и других часто используемых файлов, например журналов веб-сервера и почтовых буферов. Это приводит к уменьшению объема общедоступной памяти, но значительно ускоряет чтение и запись временных файлов. Как правило, такой подход оказывается весьма эффективным.

🕮 Подробнее о командах **lsof** и **fuser** см. в разделе 6.2.

Команда **lsof**, которая выводит список открытых файлов, и команда **fuser**, которая перечисляет процессы, использующие файловую систему, могут оказаться весьма полезными для выявления проблем, связанных с операциями обмена с диском. Эти команды отображают взаимодействия между процессами и файловыми системами и могут указать на непредусмотренные вами взаимосвязи. Например, если некоторое приложение регистрирует свои события на то же устройство, которое используется и журналами регистрации базы данных, то в результате этот диск может стать "узким местом" системы.

Утилита xdd: анализ производительности дисковой подсистемы

Современные системы хранения информации могут включать сети или SAN-элементы, RAID-массивы и другие уровни абстракции. Для измерения и оптимизации этих сложных систем имеет смысл воспользоваться утилитой **xdd**, которая доступна под открытой лицензией (GPL) и выполняется во всех наших примерах систем (не говоря уже о Windows).

Утилита **xdd** измеряет параметры подсистемы ввода-вывода на отдельном компьютере и на кластерах систем. Она хорошо описана и обеспечивает точные и воспроизводимые измерения производительности. Подробнее о ней см. на сайте `ioperformance.com`.

Команда sar: сбор статистических данных и генерирование отчетов по ним

Команда **sar** — предназначенный для мониторинга производительности инструмент, который "проверен временем" (эта команда появилась еще во времена AT&T UNIX) на системах UNIX и Linux и все еще остается актуальным, несмотря на использование "старого-доброго" синтаксиса командной строки.

На первый взгляд может показаться, что команда **sar** отображает ту же информацию, что и команды **vmstat** и **iostat**. Однако имеется одно очень важное отличие: **sar** "умеет" предоставлять отчеты как по старым (накопленным), так и текущим данным.

🕮 Пакет Linux, который содержит команду **sar**, называется **sysstat**.

Без опций команда **sar** предоставляет отчет о том, как ЦП использовался в течение того или иного дня каждые 10 минут, начиная с полуночи. Получение такой коллекции накопленных данных делает возможным сценарий **sal**, который является частью паке-

та **sar** и требует указания интервала времени, через который он должен запускаться из демона **cron**. Все данные, которые собирает команда **sar**, она сохраняет в бинарном формате в каталоге **/var/log**.

```
linux$ sar
Linux 2.6.18-92.ELsmp (bajafur.atrust.com)  01/16/2010

12:00:01  AM  CPU  %user  %nice  %system  %iowait   %idle
12:10:01  AM  all   .10   0.00    0.04    0.06   99.81
12:20:01  AM  all  0.04   0.00    0.03    0.05   99.88
12:30:01  AM  all  0.04   0.00    0.03    0.04   99.89
12:40:01  AM  all  0.09   0.00    0.03    0.05   99.83
12:50:01  AM  all  0.04   0.00    0.03    0.04   99.88
01:00:01  AM  all  0.05   0.00    0.03    0.04   99.88
```

Помимо информации о ЦП, **sar** также может предоставлять отчеты по метрическим показателям, таким как показатели дисковой и сетевой активности. Команда **sar -d** отображает сводку данных об активности диска за сегодняшний день, **sar -n DEV** — статистические данные о сетевом интерфейсе, а **sar -A** — всю доступную информацию.

Команда **sar** имеет некоторые ограничения и потому хорошо подходит только для быстрого получения кратких накопленных сведений. Тем, кто решил "всерьез и надолго" заняться мониторингом производительности, лучше установить специальную платформу с возможностями сбора коллекций данных и представления их в виде графиков, такую как платформа Cacti. Эта платформа "пришла к нам" из мира сетевого управления, но действительно умеет отображать произвольные метрические показатели системы, такие как показатели использования ЦП и памяти, в виде графиков. Пример отображаемого Cacti графика с дополнительными комментариями приводился в разделе 21.12.

nmon и nmon_analyser: мониторинг производительности в системе AIX

В системах AIX утилиту `nmon` системные администраторы часто выбирают в качестве инструмента измерения и настройки производительности. Она во многом напоминает утилиту **sar**. Стивен Аткинс (Stephen Atkins), сотрудник компании IBM, разработал крупноформатную электронную таблицу **nmon_analyser**, которая обрабатывает данные, собираемые утилитой **nmon**. Данная утилита позволяет специалистам по производительности просматривать данные в формате большой таблицы, находить "плохие" данные и составлять графики, которые можно показать клиентам. Она выполняет более сложный анализ, чем утилита **sar**. Например, она вычисляет взвешенные средние числа для анализа пиковых точек, отображает загруженность устройства, размер считанных и записанных данных за день для системы хранения IBM и строит отдельные диаграммы для систем хранения EMC. И хотя утилита **nmon_analyser** официально не поддерживается компанией IBM, вы можете найти ее на сайте IBM:

`ibm.com/developerworks/aix/library/au-nmon_analyser`.

Выбор Linux-планировщика ввода-вывода

В системах Linux используется алгоритм планирования подсистемы ввода-вывода, который выступает в роли "арбитра" между соревнующимися за дисковый ввод-вывод процессами. Он оптимизирует порядок и время запросов

для обеспечения наилучшей возможной производительности подсистемы ввода-вывода для данного приложения или ситуации.

В ядро Linux 2.6 встроены четыре разных алгоритма планирования. Можно выбирать любой. К сожалению, алгоритм планирования устанавливается во время загрузки (при помощи атрибута ядра *elevator=алгоритм*) и поэтому его не легко изменить. Алгоритм планирования системы обычно указывается в файле конфигурации начального загрузчика GRUB **grub.config**.

Все доступные алгоритмы перечислены ниже.

Comletely Fair Queuing (**elevator=cfq**). Этот алгоритм используется по умолчанию и обычно является наиболее подходящим вариантом для серверов общего назначения. Он старается равномерно предоставлять доступ к подсистеме ввода-вывода. (Во всяком случае этот алгоритм заслуживает награды за маркетинг: кто скажет "нет" совершенно справедливому планировщику?)

Deadline (**elevator=deadline**). Этот алгоритм "старается" сводить к минимуму время задержки для каждого запроса. Он изменяет порядок запросов для повышения производительности.

NOOP (**elevator=noop**). Этот алгоритм реализует простую очередь типа FIFO (First In, First Out — первым пришел, первым обслужен). Он предполагает, что запросы к подсистеме ввода-вывода уже были (или еще будут) оптимизированы или переупорядочены драйвером или устройством (каковым вполне может быть какой-нибудь контроллер со встроенной логикой). Он может быть наиболее подходящим вариантом в некоторых SAN-средах и для SSD-устройств.

Определив, какой из этих алгоритмов планирования является наиболее подходящим для данной среды (что может потребовать перепробовать все четыре варианта), возможно, вам удастся улучшить производительность подсистемы ввода-вывода.

Программа oprofile: универсальный профилировщик системы Linux

Программа **oprofile** — это чрезвычайно мощная, интегрируемая в систему программа профилирования для систем Linux с ядром версии 2.6 или выше. Профилированию могут подвергаться все компоненты системы Linux: аппаратные и программные обработчики прерываний, модули ядра, само ядро, совместно используемые библиотеки и приложения.

При наличии свободного времени и желания точно знать, как используются ресурсы системы (до самых мельчайших деталей), стоит рассмотреть вариант установки **oprofile**. Эта утилита особенно полезна для тех, кто разрабатывает свои собственные приложения или код ядра.

В состав дистрибутива **oprofile**, который доступен для загрузки на сайте `sourceforge.net`, входят как модули ядра, так и ряд пользовательских утилит.

В начале 2010 года на горизонте появилась новая система отслеживания производительности. Известная как подсистема событий производительности ("perf events"), она обеспечивает уровень оснащенности, никогда ранее не виданный в ядре Linux. По всей вероятности, эта система в будущем заменит программу профилирования производительности в системах Linux **oprofile**.

29.5. Помогите! Моя система почти остановилась!

В предыдущих разделах речь шла, в основном, о средней производительности системы. Для ее повышения требуется корректировать конфигурационные параметры или модернизировать систему.

Но даже правильно сконфигурированные системы иногда работают медленнее обычного. К счастью, нерегулярные проблемы обычно легко диагностируются. В большинстве случаев они создаются "ненасытным" процессом, который потребляет так много ресурсов процессора, жесткого диска или сетевой подсистемы, что остальные процессы буквально останавливаются. Иногда процесс намеренно захватывает ресурсы, чтобы замедлить работу системы или сети. Это называется атакой типа "отказ в обслуживании".

Очень часто, чтобы определить, какой конкретно ресурс исчерпан, не нужно даже запускать диагностическую программу. Если система начинает "подвисать"[5] или подозрительно долго обращаться к диску, то проблема, скорее всего, связана с пропускной способностью дисковой подсистемы или дефицитом памяти. Если же производительность приложений падает ниже критического уровня, проблема, возможно, в загрузке ЦП.

Первый шаг в диагностировании проблемы — запуск команды **ps auxww** (**ps -elf** в Solaris и HP-UX) или **top** для выявления явно неуправляемых процессов. Любой процесс, отнимающий более 50% времени ЦП, можно с большой долей вероятности считать ненормальным. Если столь непомерную долю ресурсов ЦП не получает ни один процесс, посмотрите, сколько процессов получают минимум по 10%. Когда их больше двух-трех (не считая самой команды **ps**), средняя загруженность, скорее всего, будет очень высокой. Такая ситуация сама по себе является причиной низкой производительности. Проверьте среднюю загруженность системы с помощью команды **uptime**, а затем выполните команду **vmstat** или **top**, чтобы узнать, простаивает ли когда-нибудь процессор.

Если конкуренции за право использования центрального процессора не наблюдается, выполните команду **vmstat**, чтобы посмотреть, какова интенсивность страничного обмена. Следует обращать внимание на все операции обмена с диском: множество операций выгрузки может означать соперничество за память, а наличие дискового трафика в отсутствие страничного обмена говорит о том, что какой-то процесс монополизировал диск, непрерывно читая и записывая файлы.

Нельзя непосредственно сопоставить дисковые операции с выполняющими их процессами, но с помощью команды **ps** можно сузить круг "подозреваемых". Любой процесс, работающий с диском, должен отнимать какую-то часть времени ЦП. Как правило, всегда можно сделать обоснованное предположение о том, какой конкретно из активных процессов захватил ресурсы[6]. Для проверки своей теории на практике выполните команду **kill -STOP**.

Предположим, процесс-виновник найден. Что с ним делать дальше? Как правило, ничего. Некоторые операции действительно требуют много ресурсов и неизбежно замедляют работу системы, но это вовсе не означает, что они незаконны. Но с помощью

[5] Другими словами, требуется больше времени на переключение между приложениями, хотя скорость выполнения отдельных заданий приемлема.

[6] Ранее признаками этого служили большой размер области данных процесса, размещенной в виртуальной памяти, либо неестественно большой объем занимаемой процессом физической памяти, но появление совместно используемых библиотек сделало эти показатели не столь полезными. Команда *ps* не умеет отделять общесистемные совместно используемые библиотеки от адресного пространства отдельных процессов. Кажется, что многие процессы занимают десятки мегабайтов физической памяти, хотя на самом деле это не так.

команды **renice** можно изменить приоритет процесса, для которого ограничивающим фактором является быстродействие ЦП. Иногда правильная настройка приложения может привести к значительному снижению потребления ресурсов. Этот эффект особенно заметен в сетевых серверных программах, например в веб-приложениях.

Процессы, интенсивно использующие диск и память, нелегко поддаются воздействию. Команда **renice** обычно не помогает. Можно уничтожить или остановить процесс, но, на наш взгляд, такая мера оправдана только в экстренных ситуациях. Лучше прибегнуть к административной мере: попросить владельца запустить процесс позже.

Ядро позволяет процессу ограничивать объем потребляемой им самим физической памяти при помощи системного вызова **setrlimit**[7]. Эта возможность также доступна в оболочке C в виде встроенной команды **limit**. Например, команда

```
% limit memoryuse 32m
```

ограничивает использование физической памяти для всех последующих пользовательских команд тридцатью двумя мегабайтами (в системе Solaris вместо команды **memoryuse** используется команда **memorysize**). Ее действие примерно эквивалентно действию команды **renice** в отношении процессов, ограничивающим фактором для которых является объем памяти.

Если неуправляемый процесс не является источником снижения производительности, нужно проанализировать две другие возможные причины. Первая — перегрузка сети. Многие программы настолько тесно связаны с сетью, что иногда трудно понять, где речь идет о производительности системы, а где — о производительности сети. Подробная информация о средствах контроля над работой сети приведена в главе 21.

В некоторых случаях проблемы, связанные с перегрузкой сети, выявить сложно, потому что они возникают и исчезают очень быстро. Например, если на всех компьютерах сети каждый день в определенное время с помощью демона **cron** запускается какая-то сетевая программа, то в сети может возникать кратковременный, но серьезный затор. Все компьютеры "зависнут" секунд на пять, после чего ситуация нормализуется.

Вторая причина — задержки, связанные с серверами. UNIX- и Linux-системы постоянно обращаются к удаленным серверам NFS, Kerberos, DNS и т.п. Если сервер не работает или какая-то иная проблема привела к тому, что связь с ним стала ненадежной, это ощущают на себе все клиентские системы.

Предположим, что в интенсивно эксплуатируемой системе какой-то процесс каждые несколько секунд вызывает библиотечную функцию **gethostent**. Если сбой в работе DNS заставляет эту функцию тратить на свое выполнение две секунды, будет заметно общее снижение производительности системы. На удивление, большое число проблем с производительностью серверов связано с неправильной конфигурацией прямого и обратного поиска в DNS.

29.6. Рекомендуемая литература

- Cockcroft Adrian and Bill Walker. *Capacity Planning for Internet Services*. Upper Saddle River, NJ: Prentice Hall, 2001.
- Drepper Ulrich. *What Every Programmer Should Know about Memory*. `lwn.net/Articles/250967`.

[7] Более тонкое управление ресурсами может быть достигнуто посредством CKRM-функций (Class-based Kernel Resource Management — управление ресурсами ядра на базе классов); см. *ckrm.sourceforge.net*.

- Ezolt Phillip G. *Optimizing Linux Performance*. Upper Saddle River, NJ: Prentice Hall PTR, 2005.
- Johnson S., et al. *Performance Tuning for Linux Servers*. Indianapolis, IN: IBM Press, 2005.
- Loukides, Mike and Gian-Paolo D. Musumeci. *System Performance Tuning (2nd Edition)*. Sebastopol, CA: O'Reilly & Associates, 2002.
- Tufte, Edward R. *The Visual Display of Quantitative Information (2nd Edition)*. Cheshire, CT: Graphics Press, 2001.

29.7. Упражнения

29.1. Определите возможную причину проблем в каждом из перечисленных ниже случаев.

а) при переключении между приложениями возникает заметная пауза и слышно, как диск интенсивно перекачивает данные;

б) программа численного моделирования выполняется дольше, чем нужно, но системная память практически свободна;

в) пользователи интенсивно эксплуатируемой ЛВС жалуются на медленный доступ через NFS, но показатель средней загруженности сервера остается очень низким;

г) В процессе работы команда часто сообщает о нехватке памяти.

29.2. ☆ Системы балансирования нагрузки способны существенно влиять на производительность серверов. Перечислите несколько механизмов, которые могут использоваться для распределения нагрузки.

29.3. ☆ Перечислите четыре основных фактора, влияющих на производительность. Для каждого из них приведите пример приложения, способного легко захватить весь ресурс. Укажите способы решения проблемы в каждом конкретном случае.

29.4. ☆ Напишите три простые программы (или сценарии). Первая должна увеличить процессорное время, выделяемое на выполнение системных задач, а вторая — на выполнение пользовательских задач. Третья не должна влиять ни на один из этих параметров, но должна иметь большую продолжительность выполнения (elapsed time). Используйте свои программы в сочетании с командами, описанными в разделе "Анализ использования центрального процессора" (выше в этой главе), чтобы узнать, что произойдет при воздействии на систему различными способами.

29.5. ☆ Напишите две простые программы (или сценарии). Первая должна интенсивно выполнять операции чтения, а вторая — операции записи. Используйте свои программы с командами, приведенными в разделе "Анализ операций обмена с диском" (выше в этой главе), чтобы узнать, что произойдет при воздействии на систему различными способами. (В качестве дополнительного условия предложите каждой из своих программ по выбору использовать модель либо произвольного, либо последовательного доступа.)

29.6. ☆ Выберите любые две программы, потребляющие заметный объем системных ресурсов. Воспользуйтесь командой **`vmstat`** и другими рассмотренными в этой главе средствами для анализа работы каждой программы. Укажите, какие действия программы приводят к повышенному потреблению ресурсов. Подтвердите свои выводы конкретными числами.