Исследование процессора, памяти и операций ввода-вывода с помощью top

Утилита top(1) предоставляет хороший обзор состояния системы, отображая информацию об использовании процессора, памяти и диска. Достаточно ввести команду top, чтобы получить полноэкранное отображение сведений о производительности системы. Информация обновляется каждые две секунды, что позволяет представить состояние системы практически в режиме реального времени.

Довольно плотно, не так ли? top пытается втиснуть как можно больше данных в стандартное окно терминала 80425 символов. Остановимся на этом выводе и объясним смысл каждой записи.

Значения PID

Каждый процесс в машине UNIX имеет уникальный идентификатор процесса, или PID. Когда бы ни стартовал процесс, ему назначается PID, значение которого на единицу превышает PID предыдущего процесса. Поле last pid — это идентификатор процесса, запущенного в системе последним. В примере в этом поле видим значение 977 (1). Следующий запущенный процесс получит идентификатор 978, затем 979 и т. д. Наблюдение за этим числом позволит увидеть, как быстро изменяется система. Если число last pid растет слишком быстро, это может свидетельствовать о выходе из-под контроля* какого-то ветвящегося процесса* или об аварийном состоянии того или иного демона.

Средняя нагрузка

Поле load average (средняя нагрузка) (2) — это не совсем понятное число. Его назначение — предоставить приблизительные данные о нагрузке на процессор системы. Средняя нагрузка равна

среднему числу потоков, ожидающих выделения времени процессора. (В других операционных системах средняя нагрузка вычисляется другими методами.) Приемлемое значение средней нагрузки зависит от системы. Если оно чрезмерно, значит, надо исследовать работу системы. Многие машины класса Pentium сталкиваются с трудностями при средней нагрузке, равной 3, а некоторые современные системы быстро работают при средней нагрузке, равной 10.

top(1) выдает три значения средней нагрузки. Первое (1.14 в нашем примере) — это средняя нагрузка за последнюю минуту. Второе (0.80) — за последние 5 минут, а третье (0.18)- за предыдущие 15 минут. Если средняя нагрузка за последние 15 минут высока, а за последнюю минуту — низка, то налицо высокий пик активности системы, который пришелся на этот 15-минутный интервал. С другой стороны, если 15-минутное значение невелико, а нагрузка за последнюю минуту высока, значит, что-то произошло за последние 60 секунд и, возможно, продолжается сейчас. Если велики все значения средней нагрузки, то такое состояние удерживалось все 15 минут.

Uptime

Последнее поле в первой строке — uptime (3). Оно сообщает о том, как долго работает система. В нашем примере система работает 18 минут, а текущее время — 18:30:34. Предоставляю вам возможность вычислить, когда произошла загрузка системы.

Количество процессов

Во второй строке (4) представлена информация о процессах, запущенных в системе в настоящее время. Работающие (running) процессы непосредственно выполняют работу; они отвечают на запросы пользователей, обрабатывают почту и выполняют все другие операции. Спящие (sleeping) процессы ожидают входных данных от того или иного источника. Это хорошие состояния процессов. Процессы в других состояниях обычно ожидают доступности ресурсов или в некотором роде зависли. Большое

количество неспящих, неработающих процессов может свидетельствовать о неполадках. Команда ps(1) покажет состояние всех процессов.

Типы процессов

Строка CPU states (5) сообщает, какая доля доступного времени процессора (в процентах) расходуется на обслуживание процессов различных типов. В этой строке представлены пять различных типов процессов: user, nice, system, interrupt и idle.

Пользовательские процессы (user) — это средние повседневные программы. Это могут быть демоны, запущенные от имени root, команды, запущенные обычными пользователями, или что-нибудь другое. Если процесс представлен в выводе команды ps -ax, это пользовательский процесс.

Процессы nice — это процессы, приоритеты которых может определять пользователь. Более подробно они рассмотрены в разделе «Изменение приоритетов с помощью nice» этой главы.

Значение system дает общее время процессора (в процентах), потраченное процессами ядра FreeBSD и пользовательскими процессами при выполнении в пространстве ядра. В состав таких процессов входят обработчики виртуальной памяти, сетевые процессы, процессы записи на диск, отладка с параметрами INVARIANTS и WITNESS и т. д.

Значение interrupt показывает, сколько времени система затрачивает на обработку запросов прерывания (IRQ).

Наконец, поле бездействующих процессов (idle) показывает, сколько времени система ничего не делает. Если время бездействия процессора очень мало, то стоит задуматься о том, чтобы пересмотреть порядок планирования заданий, или о покупке более быстрого процессора.
top и SMP

При работе с многопроцессорной системой следует иметь в виду,

что top(1) отображает средние показатели нагрузки по всем процессорам. Один процессор может быть полностью занят компиляцией, а другой может бездействовать, и в этом случае top(1) покажет, что система загружена на 50%.

Память

Далее идет строка Mem (6), представляющая действительную физическую память. Операционная система FreeBSD делит информацию об использовании памяти на несколько различных категорий.

Активная память (Active) — это общая емкость памяти, которая в данный момент задействована для выполнения пользовательских программ. Когда выполнение программы завершается, использованная ею память попадает в неактивную память (Inact). Данные, полученные с диска, помещаются в кэш (Cache). Если системе потребуется вновь запустить эту же программу, она будет взята из кэша, а не с диска.

Аналогично, запись Buf показывает емкость буфера памяти. Этот буфер содержит данные, недавно полученные с диска. Категория Buf фактически является подмножеством категорий активной памяти, неактивной памяти и кэша, а не отдельной, самостоятельной категорией.

Свободная память (Free) вообще не задействована. Это может быть память, к которой еще не было обращений, или память, освобожденная процессом. В этой системе присутствует 1 607 Мбайт свободной физической памяти. Если по истечении месяца на сервере еще остается свободная память, можно подумать о том, чтобы перенести часть памяти на машину, где ощущается ее нехватка. Результаты работы top(1) в этом примере были получены на моем ноутбуке, и ему необходим каждый бит памяти, которую он сможет получить.

В процессе поддержки пула доступной памяти операционная система FreeBSD постоянно тасует память между категориями активной, неактивной памяти и кэшем. Память в кэше легко может

перейти в категорию свободной памяти. Когда начинает ощущаться нехватка памяти в кэше, а потребность в свободной памяти остается высокой, FreeBSD выбирает страницы из пула неактивной памяти, проверяет, могут ли они использоваться в качестве свободной памяти, и перемещает их в пул свободной памяти. FreeBSD старается сохранить общее число страниц свободной памяти и кэша выше значения sysctl vm.v_free_target. (Размер страницы определяется параметром sysctl hw.pagesize, значение которого для платформ i386 и amd64 составляет 4 096 байт.)

Наличие свободной памяти в системе вовсе не означает, что в системе достаточно памяти. Если vmstat(8) показывает большой объем операций с пространством свопинга, это свидетельствует о нехватке памяти. В системе может работать программа, которая постоянно освобождает память. Кроме того, FreeBSD перемещает некоторые страницы из кэша в пул свободной памяти, чтобы поддержать заданный уровень свободной памяти.

Связанная память (Wired) — это память, применяемая для структур данных ядра, а также для особых системных вызовов, которым память нужна немедленно. Связанная память никогда не вытесняется в пространство свопинга.

Своп

Далее идет строка Swap (7), содержащая данные о доступном пространстве свопинга (области подкачки) и о ее задействованном объеме. При свопинге (swapping) система использует дисковый накопитель как дополнительную память. Далее в этой главе пространство свопинга рассмотрено более подробно.

Список процессов

Наконец, top(1) выводит список процессов системы и их базовые характеристики. Формат таблицы подразумевает вывод как можно большего объема информации в максимально сжатой форме. Каждый процесс описан в отдельной строке.

PID
Первая колонка — числовой идентификатор процесса, или PID. Каждый процесс, запущенный в системе, имеет уникальный PID. Выполняя команду kill(1), вы указываете процесс по его PID. (Если PID процесса неизвестен, можно использовать команду pkill(1), чтобы указать процесс по его имени.)

USERNAME
Следующая колонка — имя пользователя, от имени которого запущен процесс. Если несколько процессов с одним пользовательским ID интенсивно потребляют время процессора или память, вы знаете, с кем поговорить.

PRI и NICE
Колонки PRI (priority — приоритет) и NICE взаимосвязаны. Они показывают, какой приоритет система назначает тому или иному процессу. О приоритетах и nice речь пойдет чуть далее.

SIZE
Это объем памяти, которую система выделила для этого процесса.

RES
В колонке RES (Resident Memory) показано, какая часть программы действительно находится в памяти (резидентна). Для программы может быть отведен огромный объем памяти, но задействована ею будет лишь малая его часть. Например, в то время, как я пишу эти строки, моей программе OpenOffice.org выделено 169 Мбайт, но используется только 103 Мбайт. (Интересное наблюдение: в память, используемую X.org, включается память видеокарты, которая не входит в состав физической памяти системы.)

STATE
В колонке STATE показано, что делает процесс в данный момент. Процессы могут пребывать в разных состояниях: ожидать ввода данных; спать, пока их кто-нибудь не разбудит; выполняться и т. д. Здесь можно видеть имя системного вызова, такого как select, pause или ttyin, в котором процесс ожидает наступления

определенного события. В многопроцессорных системах для работающих процессов можно видеть, на каком процессоре он исполняется. В нашем примере процесс acroread исполняется на процессоре CPU0.

TIME
Колонка TIME сообщает, какой объем процессорного времени использовал процесс.

WCPU
Колонка WCPU (использование CPU) выдает взвешенные показатели использования CPU, а именно процентное отношение времени CPU, которое процесс использует согласно своему приоритету и значению nice.

COMMAND
Наконец, в колонке COMMAND представлено имя программы.

Вывод команды top(1) позволяет понять, где система тратит больше всего времени.
top(1) и ввод/вывод.

Помимо стандартной статистики использования процессора, у top(1) есть режим вывода информации об операциях ввода-вывода, в котором отображаются процессы, наиболее активно использующие диск. Чтобы перейти в режим отображения информации об операциях ввода-вывода, нужно после запуска top(1) нажать клавишу M. В верхней части экрана по-прежнему выводится информация об использовании памяти, пространства свопинга и о состоянии процессора, но нижняя часть существенно изменилась.

Колонка PID — это идентификатор процесса, а в колонке USERNAME выводится имя пользователя, от лица которого был запущен процесс.

Название колонки VCSW происходит от voluntary context switches (добровольное переключение контекста) и показывает, сколько раз процесс добровольно уступил систему другим процессам. Название колонки IVCSW означает involuntary context switches

(принудительное переключение контекста) и показывает, сколько раз ядро сообщало процессу: «Ваше время истекло, пора дать поработать другим процессам».

Аналогично, в колонках READ и WRITE выводится число дисковых операций чтения и записи, произведенных системой. В колонке FAULT показано, как часто этот процесс перемещал страницы памяти из пространства свопинга, что по сути является одной из разновидностей дисковых операций. Сумма значений этих трех столбцов выводится в колонке TOTAL.

В колонке PERCENT показан процент дисковых операций, приходящийся на долю этого процесса. В отличие от gstat(8), команда top(1) выводит все значения в процентах от фактической дисковой активности, а не от максимальной возможной. Если в системе имеется единственный процесс, использующий диск, top(1) покажет, что этому процессу принадлежат все 100% дисковых операций, даже если он всего лишь записал немного данных. Команда gstat(8) показывает уровень нагрузки на диск, а top(1) — какие процессы ответственны за эту нагрузку. Здесь мы видим, что все дисковые операции выполняются процессом с идентификатором 3064. Это процесс командного интерпретатора tcsh(1). Давайте понаблюдаем за «злодеем».

Дополнительные возможности команды top

Команда top может выводить информацию на экран разными способами. С ее помощью можно просматривать процессы, принадлежащие определенному пользователю, включая или исключая потоки ядра, и т. д. За дополнительной информацией обращайтесь к странице руководства.

top -SIH =>