# Оценка загруженности компьютера в различных UNIX-системах[$]

С. А. Жуматий[*], С. И. Соболев[*]

*Статья посвящена исследованию методов определения загруженности компьютера, работающего под управлением ОС UNIX/Linux. Данное исследование было проведено при реализации модуля системы метакомпьютинга, отвечающего за запуск распределенных задач в моменты простоя доступных компьютеров. В статье анализируются различные способы определения загруженности вычислительных ресурсов, обсуждаются их достоинства и недостатки. Описывается метод, выбранный по результатам анализа.*

Многие в детстве играли в жмурки. С завязанными глазами надо было поймать кого-нибудь из играющих и определить, кто это. Для определения приходилось полагаться только на косвенные признаки — рост, элементы одежды... Как легко было ошибиться!

Очень похожая ситуация возникла при создании системы организации метакомпьютерных расчётов. Система должна отслеживать состояние распределённой вычислительной среды, состоящей из самых разных компьютеров, работающих в самых разных режимах, принадлежащим самым разным людям и организациям. На свободных компьютерах можно запускать расчёты. Если на компьютере начинает работать приложение хозяев, расчёт необходимо прекратить, чтобы не мешать штатной работе компьютера.

[$]Работа выполнена при поддержке гранта РФФИ №05-07-90206

[*]Научно-исследовательский вычислительный центр МГУ

Как корректно определить степень загрузки компьютера при том, что никаких привилегированных прав на компьютерах у нас нет? Задача и в самом деле напоминает игру в жмурки, но в отличии от игры, ошибка тут стоит дороже. Из доступных инструментов — только те системные программы, которые установлены на целевом компьютере. Основной платформой было избрано семейство ОС Linux, как наиболее распространённое в вычислительной практике, для которой мы попробуем найти методы решения задачи по определению факта загруженности компьютера.

Пользователь Windows мог бы сказать: «Решение элементарно! Есть же хранитель экрана, который запускается во время простоя компьютера — вот его и надо использовать. Система сама и запустит, и завершит его когда надо. Кстати, хранители экрана есть не только для Windows...».

Верно, но есть нюансы. Хранитель экрана запускается только во время бездействия пользователя, но не обязательно бездействия процессора. Процессор может быть занят, например, пакетной обработкой фотографий в Photoshop или преобразованием видеозаписи из одного формата в другой. А уж вычислительный сервер вообще может не иметь ни мыши, ни клавиатуры и никаких хранителей экрана в принципе. Поэтому хранители экрана нашей проблемы не решают.

Не оставляет чувство, что решение такой задачи должно быть очень простым, поскольку есть множество стандартных программ, с помощью которых можно определить степень загруженности. Но не всё так просто. Важным требованием является переносимость в рамках UNIX-систем. Однако, как мы увидим далее, даже в семействе Linux проблема не решается тривиально. Решение дополнительно осложняется тем, что на вычислительном узле у нас может не быть прав администратора, а значит, набор средств будет ограничен.

Самый «правильный» и гарантированно работающий метод — использовать протокол и инфраструктуру SNMP. Но далеко не на каждой рабочей станции установлено программное обеспечение, реализующее серверную часть SNMP. Так как рабочая станция или сервер могут быть «чужими» и прав суперпользователя мы там мо-

жем не иметь, то устанавливать своё системное программное обеспечение администратор может и не разрешить.

Аналогичная ситуация сложилась с использованием программ пакета SAR. Представление данных в этом пакете фиксировано, но он не является стандартным и установлен далеко не на всех системах. На каждой ОС существует независимая реализация этого пакета, поэтому даже использовать исходные тексты не всегда возможно.

Остаётся использование стандартных программ, входящих в системные пакеты, таких как ps, top, uptime.

Самым логичным решением казалось использование программы ps, выдающей загрузку процессора отдельно по каждому процессу. Программа ps вызывалась из скрипта, запускающегося раз в минуту с помощью cron. Её вывод анализировался, отдельно суммировались проценты загрузки процессора «своими» и чужими процессами, и на основании этих данных делался вывод о необходимости запуска либо снятия метакомпьютерной задачи.

В ходе экспериментов выяснилось, что ps далеко не всегда корректно выдает сведения о потребление процессорных ресурсов. Наблюдались случаи, в которых на компьютерах с запущенной счетной программой ps демонстрировала нулевой процент загрузки активными процессами, при этом другие средства мониторинга (top) указывали, что загрузка компьютера этими процессами близка к 100%. Очевидно, это ошибка или некорректность в самой программе ps. Таким образом, данные, предоставляемые программой ps, невозможно использовать для решения поставленной задачи.

Следующим вариантом реализации определения загрузки компьютера был скрипт, анализирующий выдачу команды top, доступной на большинстве UNIX-платформ. Традиционно эта программа используется в интерактивной форме для непосредственного наблюдения, однако в ней предусмотрен режим для пакетного запуска.

Мы запускали команду в следующем виде: top -b -n 1 (пакетный режим, одна итерация). Но и от этого варианта пришлось отказаться. Во-первых, в начале запуска сама программа top достаточно сильно загружает ОС, и данные о загрузке оказываются необъективными. Во-вторых, формат выдачи результатов сильно зависит

от версии команды и дистрибутива. В двух дистрибутивах Linux (RedHat Linux 7.3 и SuSE Linux 10.0), установленных на узлах суперкомпьютерного комплекса НИВЦ МГУ, версии, а точнее, ветки развития команды top оказались принципиально разными.

Вот пример части экрана, получаемого на разных дистрибутивах:

**RedHat 7.3**

4:59pm up 321 days, 6:43, 18 users, load average: 0.09, 0.23, 0.19
292 processes: 264 sleeping, 1 running, 27 zombie, 0 stopped
CPU0 states: 50.0% user, 50.0% system, 50.0% nice, 0.0% idle
CPU1 states: 18.0% user, 81.0% system, 9.0% nice, 0.0% idle
Mem: 1033296K av, 1019892K used, 13404K free, 0K shrd, 51908K buff
Swap: 530104K av, 127548K used, 402556K free 620732K cached
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
22243 root 15 0 1156 1156 804 R 8.2 0.1 0:00 top
1 root 9 0 412 376 360 S 0.0 0.0 5:51 init

**SuSE 10.0**

top - 17:01:31 up 174 days, 5:21, 11 users, load average: 0.27, 0.27, 0.15
Tasks: 253 total, 2 running, 247 sleeping, 0 stopped, 4 zombie
Cpu0 : 0.3% us, 2.3% sy, 0.0% ni, 96.0% id, 0.0% wa, 0.0% hi, 1.3% si
Cpu1 : 0.0% us, 0.0% sy, 0.0% ni, 100.0% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 2050860k total, 1405736k used, 645124k free, 191172k buffers
Swap: 979176k total, 4056k used, 975120k free, 751500k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
25171 root 15 0 57644 37m 2132 S 0.3 1.9 11:18.15 cleo
23196 golovin 15 0 26644 2520 1940 S 0.3 0.1 3:05.63 ssh

Видно, что для человека разница небольшая, но программы разбора такой выдачи будут значительно отличаться. И что гораздо хуже — нет никакой гарантии, что не потребуется разбор и других

вариантов выдачи, о которых мы пока не знаем, но которые могут встретиться на компьютерах вычислительной среды.

Аналогичная ситуация сложилась и с программой vmstat. На различных версиях ОС она выдаёт результаты в различном формате.

**RedHat 7.3**

```
procs memory swap io system cpu
r b w swpd free buff cache si so bi bo in cs us sy id
0 0 0 130848 14292 148368 441620 0 0 2 2 1 2 0 3 2
```

**SuSE 10.0**

```
procs ————memory————- —swap– ——io—- –system– —-cpu—-
r b swpd free buff cache si so bi bo in cs us sy id wa
0 0 134204 62872 238056 232792 0 1 1 3 5 6 11 4 80 4
```

Самой «стандартной» программой оказалась команда uptime, которая выдаёт значение loadaverage узла. Её вывод на большинстве UNIX-платформ практически единообразен и приемлем для программного анализа. Величина loadaverage равна среднему числу процессов, ожидающих квант процессорного времени в очереди. Например, если в данный момент на компьютере запущено 5 счётных задач, то loadaverage будет не меньше 5. Команда uptime при запуске выдает три значения loadaverage: за последнюю минуту, 5 минут и 15 минут.

Однако высокое значение loadaverage, на которое можно было бы ориентироваться, не обязательно означает высокую загрузку процессора. К примеру, задача может ожидать завершения системного вызова, такого как запись на жёсткий диск. При этом она будет находиться в очереди, и loadaverage увеличится на единицу, хотя реальная загрузка процессора может быть очень низкой.

Возможна ситуация, когда средняя загрузка возрастает только за счет системных задач, и в этом случае прикладное счетное приложение также снимается со счета. Обратная ситуация исключается, т.е. низкий уровень loadaverage гарантирует, что процессор не занят никакими задачами. Эмпирическим путем были установлены

следующие пороги, по достижению которых узел считается «занятым» или «свободным» (Ncpu — общее число процессоров на узле, loadaverage — средняя загрузка узла за последнюю минуту).

- «свободен», loadaverage < Ncpu*0.25 (предполагаем, что 25% процессорного времени может требоваться системным процессам);
- «занят», loadaverage > Ncpu + 0.9 (запуская столько процессов прикладной задачи, сколько процессоров на узле, проверяем, не появился ли еще хотя бы один активный процесс).

Так как значение loadaverage не «мгновенное», а усреднённое, то вероятность запуска задачи в момент кратковременного простоя компьютера снижается. По этой же причине определить факт возможности запуска счётной задачи удаётся только по прошествии какого-то времени.

Таким образом, решение поставленной задачи крайне осложнено тем, что общедоступный набор системных программ, во-первых, не всегда соответствует стандартам, и, во-вторых, не всегда выдаёт адекватную информацию.

Если ставить задачу достаточно широко, а именно определение загруженности компьютера в рамках наиболее распространённых ОС семейства UNIX, то, с нашей точки зрения, её решение практически невозможно. Это связано с тем, что не удаётся найтии работающего одинаково на всех UNIX-платформах штатного средства, посредством которого можно было бы определять загрузку процессора, даже если у нас есть права суперпользователя.

Если ограничить задачу только ОС Linux, то решение затруднено, но возможно. Вероятное решение — использовать данные из файловой системы /proc. Эти данные заведомо актуальны и формат необходимых файлов не меняется в зависимости от дистрибутива. Препятствием может быть усиленная система безопасности, например gr-security или se-linux, но в этом случае получение необходимых нам данных будет практически невозможно.

Проблема с чтением данных из /proc состоит в том, что объём информации довольно велик — надо анализировать столько фай-

лов, сколько сейчас существует процессов в системе. Это десятки, а иногда и сотни файлов. Их анализ в скрипте занимает значительное время, а значит, общая картина будет сильно искажаться. Можно анализировать эти данные в программе, написанной, например, на Си. Но в этом случае необходимо компилировать эту программу перед установкой в системе, так как бинарный файл может не работать на разных архитектурах, а иногда и на одной и той же архитектуре, но с разными версиями ОС. Компиляция же вспомогательного файла нежелательна, ведь на компьютере вычислительной среды может не быть компилятора.

В конечном итоге мы остановились на варианте использования программы uptime. Это решение не всегда даёт корректные результаты. Можно получить ответ «система занята» при том, что процессор простаивает. Но нельзя получить ответ «система свободна», если процессор реально занят счётными задачами. Это значит, что используя такой подход, можно «прозевать» время, когда процессор простаивает, но нельзя запустить задачу, если процессор занят.

Интенсивное тестирование скрипта определения занятости серверов на узлах суперкомпьютерного комплекса НИВЦ МГУ (запуск заданий через систему метакомпьютинга на фоне работающих штатных приложений) показало пригодность описанного алгоритма для решения поставленной задачи, с небольшой оговоркой, что иногда будет возникать «иллюзия» занятости системы. При тестировании было задействовано более 100 вычислительных узлов трёх вычислительных кластеров в течении нескольких месяцев.

Тем не менее, поиски оптимального решения продолжаются. Мы приглашаем к обсуждению специалистов, сталкивающихся с подобными задачами, присоединиться к дискуссии по определению загрузки серверов можно на форуме сайта PARALLEL.RU (http://parallel.ru/forum/).