

Мониторинг `sar`, `atop`, `dstat`

`sar` — показывает более чем достаточную информацию о необходимых нам четырех ресурсах системы, умеет «возвращаться в прошлое», но не умеет фокусироваться на процессах.

`vmstat` и `iostat` — могут мониторить CPU, память и использование диска только на системном уровне, но не на уровне процесса, также не умеют «возвращаться в прошлое».

`top` — как безусловно самая популярная утилита, мониторит CPU и память на уровне системы и на уровне процессов. Но как и многие другие показывает только текущую ситуацию, самым главным минусом на мой взгляд это то, что ее показания не всегда точны, используя утилиту `top` мы можем столкнуться с такой проблемой: система будет указывать нам что занять 90% CPU, а на самом деле сумма загруженности CPU на уровне процессов будет 40% (разница в 50% процентов указывает, что система показывает нам информацию между предыдущим и текущим снимком, хотя система в принципе уже свободна). Т.е достоверность и информативность немного хромает.

Ниже приведена таблица сравнения этих систем по характеристикам.

	<i>sar</i>	<i>vmstat</i>	<i>iostat</i>	<i>top</i>	<i>atop</i>
Live measurement	✓	✓	✓	✓	✓
Historical data	✓	✗	✗	✗	✓
System-level measurements					
Processor	✓	✓	✓	✓	✓
Memory	✓	✓	✗	✓	✓
Disk	✓	✗	✓	✗	✓
Network	✓	✗	✗	✗	✓
Process-level measurements					
Processor	✗	✗	✗	✓	✓
Memory	✗	✗	✗	✓	✓
Disk	✗	✗	✗	✗	✓
Network	✗	✗	✗	✗	✓ with patch

Утилита atop по умолчанию умеет показывать загрузку по процессору, памяти, дискам и сети. Также доступен патч, который будет показывать использование сети по процессам. По умолчанию стандартным выводом команды является следующая информация:

Как и в стандартном выводе top экран разделен на две половины, в первой из них указаны данные о системе, а во второй данные по процессам.

В верхней половине видна не только информация по процессору и по памяти как в стандартном выводе top, а еще информация касательно сети и дисков. Если мы обратим внимание на 2 строку данного вывода, просуммировав значения (3+2+0+195+0) мы получим 200% — это означает, что система является 2 процессорной, и каждый из них имеет по 100%, ниже дается деление по каждому процессору, на сколько тот или иной занят. Ниже после этих строк указана оперативная память, информация по дискам, и сетевым интерфейсам.

Более подробное описание каждого столбца можно найти в man странице к данной утилите, кстати на редкость подробное описание.

Естественно мы можем менять ее вид так как нам захочется используя горячие клавиши, ниже я указал наиболее интересные из них,

m — выстроит процессы по занятой памяти

d — построит процессы по использованию диска

n — выстроит процессы по использованию сети (доступно только с установленным патчем)

v — покажет более подробную информацию о процессах (пользователя, дату и время начала процесса)

u — выстроит таблицу по самым прожорливым пользователям

Использование *shift+(d,m,n)* выстроит текущий список по выше указанным параметрам.

i — изменение времени проверки, по умолчанию 10 секунд.

g – вернет все в дефолтный вывод.

Теперь давайте посмотрим как все же пользоваться функцией «возврата в прошлое», у утилиты *atop* при стандартной установке есть такая полезная «фича», которая включает ночное логирование производительности системы. Утилита по умолчанию снимает показания каждые 10 минут, и сбрасывает их в лог */var/log/atop-YYYYMMDD*.

%iowait – показатель, означающий процентное соотношение времени процессора, потраченное на ожидание ввода/вывода.

Высокий *%iowait* говорит о том, что система ограничена возможностями дисковой памяти, выполняя множество операций дискового ввода-вывода, что приводит к замедлению работы системы.

uptime

Для многоядерных систем лучше смотреть число *cpu cores*.

```
$ cat /proc/cpuinfo | grep cores
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

```
cpu cores: 4
```

(как раз тот случай – 4 физических ядра, 8 «логических»).

LA может быть хорошим оповещением – если вырастает в 10 раз выше обычного – надо срочно спасать сервер пока контроль не потерял. Как правильно замечали про NFS (особенно с `-hard`) бывает и 1200 LA при отлично работающем сервере.

Так же неплохим критерием опасности является CPU idle – если среднее значение за день меньше 30% надо по этому поводу что-то предпринять.

Очень часто LA является указателем на нехватку памяти – мониторить свободную память в линуксе дело еще менее благодарное чем гадать на LA, но все вместе составляет симптом. Первейшее дело это запустить `'vmstat 1'` и смотреть колонки `r,b,si` и `so`. Если в `b` какие неадекватные цифры, а в `r` единички и в то же время в `si/so` счетчик идет на тысячи и десятки тысяч – надо прям щаз убивать какой-то жирный процесс, а то можно потерять контроль.

LA может быть высоким, но реально система будет ненагруженной. Будут просто висеть процессы со статусом D например.

Зачастую завершение этих процессов невозможно т.к. это `uninterruptible sleep (usually IO)`. Помогает перезагрузка.

Моё грубое определение Load в Linux – это число процессов, находящихся в состоянии R (`running or runnable (on run queue)`) + число процессов в состоянии D (`uninterruptible sleep (usually IO)`). Load average усредняет эти значения по хитрой формуле (кажется, это экспоненциально взвешенное скользящее среднее).

Посмотреть статусы можно через `ps`.

Статья в таком виде не даёт никакого понимания о LA в Linux.

Мало того, LA в Linux вообще не информативен, ибо смешивает очередь I/O и очередь на выполнение процессором. Как я уже говорил, смотреть нужно в `ps`.

<https://habr.com/ru/post/216827/>

```
dstat -tldnpms 10
```

При этом будет выводиться:

- текущее время — **t**
- средняя загрузка системы — **l**
- использования дисков — **d**
- загрузка сетевых устройств — **n**
- активность процессов — **p**
- использование памяти — **m**
- использование подкачки — **s**
- с интервалов в **10** секунд

Еще пример использования. Вывод самых *прожорливых* процессов:

```
dstat -tl -M top-cpu,top-io -d -M top-mem -m 10
```

```
dstat -tl -top-cpu -d -top-io -n -top-mem -m 10
```