

Linux и облачные вычисления

# Мониторинг ресурсов

Мониторинг ресурсов (оперативная память, процессоры, диски). Использование утилиты htop. Архивирование файлов. Поиск данных.

## Оглавление

[Мониторинг ресурсов](#)

[htop](#)

[Мониторинг дисков](#)

[Команда df](#)

[Команда du](#)

[Архивирование файлов](#)

[Команда tar](#)

[Поиск файлов](#)

[Команда find](#)

[Практическое задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

# Мониторинг ресурсов

Специалисту, работающему с данными, приходится регулярно следить за размером свободной оперативной памяти. Это связано с тем, что в некоторых библиотеках (например, в Pandas) наборы данных часто целиком располагаются в оперативной памяти, что позволяет ускорить работу с ними. Мониторинг ресурсов оперативной памяти необходим как при единовременных задачах, так и при создании программ, работающих на регулярной основе или в режиме реального времени — чтобы избежать их аварийного завершения.

Также необходимо следить за тем, как расходуются ресурсы процессоров. Дело в том, что многие алгоритмы машинного обучения работают в многопоточном режиме, и при одновременном использовании нескольких таких алгоритмов процессорные мощности могут расходоваться не оптимально. От этого модели машинного обучения работают медленнее. В данном уроке мы будем использовать термин «процессор» как синоним понятия «ядро».

При работе с большими объемами данных, даже если оперативная память и процессоры используются оптимизированно, могут возникать ситуации, когда на сервере образовалось большое количество данных и заканчивается место на диске. Предотвращать такие случаи также позволяет мониторинг свободного дискового пространства.

На этом уроке расскажем о том, как можно контролировать использование оперативной памяти, процессоров и дисков, работая в операционной системе Linux.

## htop

Чтобы следить за использованием оперативной памяти и процессоров, удобно пользоваться утилитой **htop** (по функциям она похожа на стандартную утилиту **top**). Она не входит в стандартный дистрибутив Ubuntu, поэтому нужно будет установить ее с помощью команды **sudo apt-get install htop**:

```
ubuntu$ sudo apt-get install htop
```

После запуска команды утилита **htop** будет установлена:

```
ubuntu$ sudo apt-get install htop
Reading package lists... Done
Building dependency tree
Reading state information... Done
htop is already the newest version (2.1.0-3).
The following package was automatically installed and is no longer required:
  unzip
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 99 not upgraded.
ubuntu$
```

Запустим утилиту с помощью команды **htop** либо **sudo htop**:

```
ubuntu$ htop
```

Появится панель **htop** с информацией об используемых ресурсах и работающих процессах:

CPU[  0.7%] Tasks: 29, 25 thr; 1 running											
Mem[      95.5M/984M] Load average: 0.00 0.00 0.00											
Swp[ OK/OK] Uptime: 00:54:59											
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2424	ubuntu	20	0	32196	4260	3680	R	0.0	0.4	0:00.28	htop
761	root	20	0	281M	6948	6080	S	0.0	0.7	0:00.05	/usr/lib/accountsservic
2385	ubuntu	20	0	105M	3280	2268	S	0.0	0.3	0:00.02	sshd: ubuntu@pts/0
924	root	20	0	545M	28644	17436	S	0.0	2.8	0:00.05	/usr/lib/snapd/snapd
771	root	20	0	545M	28644	17436	S	0.0	2.8	0:00.73	/usr/lib/snapd/snapd
1	root	20	0	156M	9200	6836	S	0.0	0.9	0:02.76	/sbin/init
384	root	19	-1	124M	14656	13956	S	0.0	1.5	0:00.33	/lib/systemd/systemd-jo
413	root	20	0	44424	5504	3272	S	0.0	0.5	0:00.33	/lib/systemd/systemd-ud
414	root	20	0	97708	1928	1752	S	0.0	0.2	0:00.00	/sbin/lvmetad -f
490	systemd-t	20	0	138M	3352	2824	S	0.0	0.3	0:00.00	/lib/systemd/systemd-ti
459	systemd-t	20	0	138M	3352	2824	S	0.0	0.3	0:00.02	/lib/systemd/systemd-ti
602	systemd-n	20	0	80024	5384	4784	S	0.0	0.5	0:00.01	/lib/systemd/systemd-ne
610	systemd-r	20	0	70616	5212	4656	S	0.0	0.5	0:00.02	/lib/systemd/systemd-re
902	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.04	/snap/amazon-ssm-agent/
903	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.00	/snap/amazon-ssm-agent/
904	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.00	/snap/amazon-ssm-agent/
911	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.02	/snap/amazon-ssm-agent/
921	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.00	/snap/amazon-ssm-agent/
928	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.00	/snap/amazon-ssm-agent/
929	root	20	0	530M	12816	11144	S	0.0	1.3	0:00.00	/snap/amazon-ssm-agent/
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit											

Здесь мы видим, что CPU используется на 0,7 %. В данном инстансе только один процессор, а если бы их было несколько, то они были бы пронумерованы по порядку. Зеленым цветом в диаграмме, показывающей загрузженность CPU, обозначается часть времени процессора, занятая процессами с нормальным приоритетом. Остальных цветов на данной диаграмме нет, но о них тоже расскажем. Синим обозначается часть, занятая процессами с низким приоритетом. Красным — процессы с приоритетом ядра. Желтым — время процессора, потраченное на виртуализацию либо невольно «украденное» другими пользователями, которые работают в других виртуальных серверах, но на тех же физических ресурсах, и активно нагружают процессоры.

Далее располагается информация об использовании оперативной памяти (**Mem**): 95,5 Мб из 984 Мб. (В данном примере используется инстанс **t2.micro**, имеющий довольно скромные ресурсы по сравнению с инстансами, которые обычно используют в рабочих условиях. Тем не менее для изучения основных принципов работы в **bash** такой инстанс вполне подходит.)

Мы видим, что диаграмма, отображающая использование оперативной памяти, состоит из трех цветов: зеленого, синего и желтого.

Зеленый цвет показывает используемую оперативную память, синий — буферы и желтый — системный кеш.

Раздел **Swp** показывает использование **swap** (своп) — в данном случае оно нулевое.

Далее располагается информация о процессах. Наиболее информативными здесь будут поля:

- **PID** — id процесса;
- **USER** — пользователь, от чьего имени запущен процесс;
- **PRI** — текущий приоритет процесса;

- **RES** — количество резидентной, то есть не перемещаемой в **swap**, памяти в килобайтах;
- **SHR** — количество разделяемой, то есть **shared**, памяти программы в килобайтах (памяти, которая может быть использована другими приложениями);
- **CPU %** — использование процессора в процентах;
- **MEM %** — использование процессом памяти в процентах;
- **TIME+** — время работы процесса;
- **Command** — команда, которой был запущен процесс.

Более подробное описание программы **htop** можно прочитать по ссылке: <http://linux-bash.ru/menusistem/79-htop.html>.


Чтобы выйти из программы, нужно нажать F10.

## Мониторинг дисков

### Команда df

Для мониторинга дисковых ресурсов можно пользоваться утилитами **df** и **du**.

Например, вот так можно в удобном виде получить информацию об использовании дискового пространства с помощью программы **df -h** (в данном случае **-h**, сокращение от слова **human**, используется для более привычного человеку отображения):



```
ubuntu$ df -h
```

После запуска программы можно увидеть название устройства (диска), общий размер, используемое пространство в абсолютном значении, доступное пространство, используемое пространство в процентах и путь, к которому прикреплено устройство.

```

ubuntu$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            481M   0    481M   0% /dev
tmpfs           99M   728K   98M    1% /run
/dev/xvda1      7.7G  2.3G  5.5G   29% /
tmpfs           492M   0    492M   0% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           492M   0    492M   0% /sys/fs/cgroup
/dev/loop0       91M   91M     0 100% /snap/core/6350
/dev/loop1       18M   18M     0 100% /snap/amazon-ssm-agent/930
/dev/loop2       88M   88M     0 100% /snap/core/5328
/dev/loop3       13M   13M     0 100% /snap/amazon-ssm-agent/495
/dev/loop4       90M   90M     0 100% /snap/core/6130
tmpfs           99M   0     99M   0% /run/user/1000
ubuntu$

```

Например, на данном скриншоте устройство **/dev/xvda1** имеет размер 7,7 Гб, из них использовано 2,3 Гб, доступно 5,5 Гб, использовано 29 %, диск закреплен за корневой директорией (“/”).

## Команда du

С помощью команды **du** по умолчанию можно узнать размеры всех файлов и папок, находящихся в текущей директории. Команда **du -h** покажет их размеры в удобном для человека формате.

Если нужно посмотреть суммарный объем файлов и папок, находящихся в текущей директории, следует запустить команду **du -s**.

Запустим команду **du -sh**, показывающую суммарный объем файлов и папок в текущей директории (в данном примере это **/home/ubuntu**):

```

ubuntu$ du -sh

```

После запуска команды можно увидеть, что суммарный объем составляет 86 Мб.

```
ubuntu$ du -sh
86M  .
ubuntu$
```

# Архивирование файлов

## Команда tar

С помощью программы **tar** можно сжимать данные (при этом выбирать разные алгоритмы сжатия информации), а также объединять несколько файлов в архив.

Чтобы создать архив под названием **archive.tar.gz**, в котором будет содержаться файл **Shakespeare.txt** (с ним мы уже работали в домашней директории пользователя **ubuntu**), запустим команду **tar -cf archive.tar.gz Shakespeare.txt**. Опция **c** (от слова **create**) в этой команде означает, что нужно создать архив, а опция **f** обозначает файл для архива:

```
ubuntu$ tar -cf archive.tar.gz Shakespeare.txt
```

Посмотрим содержимое домашней директории — там появился файл **archive.tar.gz**:

```
ubuntu$ ls
Shakespeare.txt  create_matrix.py  instrument_table.txt
archive.tar.gz   header.txt        instruments.txt
ubuntu$
```

Чтобы создать сжатый архив, нужно помимо опций **c** и **f** указать опцию **z**. И тогда команда запустится как **tar -zcf archive\_z.tar.gz Shakespeare.txt** (здесь мы создаем файл с названием **archive\_z.tar.gz**):

```
ubuntu$ tar -zcf archive_z.tar.gz Shakespeare.txt
```

Посмотрим на размер файлов, используя команду **du \*.gz**:

```
ubuntu$ du -h *.gz
12K    archive.tar.gz
4.0K    archive_z.tar.gz
ubuntu$
```

Видно, что файл **archive\_z.tar.gz** в 3 раза меньше **archive.tar.gz**.

Извлечь файлы из архива можно с помощью команды **tar -xf archive.tar.gz**. В этой команде вместо опции **c**, использованной при создании архива, применяется опция **x** (от слова **extract**).



# Поиск файлов

## Команда find

Для поиска файлов удобно использовать команду **find**. Например, можно найти все файлы с расширением **.txt** в текущей директории с помощью команды **find \*.txt**:

```
ubuntu$ find *.txt
```

Результат будет выглядеть так:

```
ubuntu$ find *.txt
Shakespeare.txt
header.txt
instrument_table.txt
instruments.txt
ubuntu$
```

## Практическое задание

1. Запустить **htop** и посмотреть, сколько процессоров и оперативной памяти есть на сервере.
2. Найти все программы с расширением **.py**.
3. \* Создать и запустить программу на Python, выводящую числа от 0 до 100 включительно.

Запустить **htop** во время выполнения программы и найти выполняемую программу в списке процессов, используя поиск по ключевому слову **python** (использовать средства поиска **htop**).

## Дополнительные материалы

1. [htop и многое другое на пальцах.](#)
2. [Поиск файлов с помощью find.](#)
3. [18 примеров команды tar в Linux.](#)

## Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [HTOP — монитор процессов.](#)
2. [htop — продвинутый консольный монитор процессов.](#)
3. [Команда tar в Linux.](#)