ОНЛАЙН-ОБРАЗОВАНИЕ

Что такое неотзывчивая система

Что такое неотзывчивая система

- Долго отвечает определенный сервис
- Определенный сервис пропускает запросы
- Система в целом любой запрос выполняет долго
- Система в принципе не отвечает ни на какой запрос

Самая базовая метрика

Самая базовая метрика
LA - Load Average

Число процессов стоящих в очереди на исполнение

```
# cat /proc/loadavg
0.13 0.05 0.13 1/89 5903
```

```
# uptime
 12:23:23 up 9 days, 23:57,  2 users,  load average: 0,11, 0,05, 0,13

# top
top - 12:24:23 up 9 days, 23:58,  2 users,  load average: 0,04, 0,04, 0,12
```

Самая базовая метрика - LA

Число процессов стоящих в очереди на исполнение

Причина высокого LA - борьба за ресурсы.

* **CPU (cpu usage/process count)**
* **Disk (IO/SWAP)**
* **Сеть**

CPU Usage

Борьба за ресурс CPU

Cpu usage большой, LA большой, мало процессов

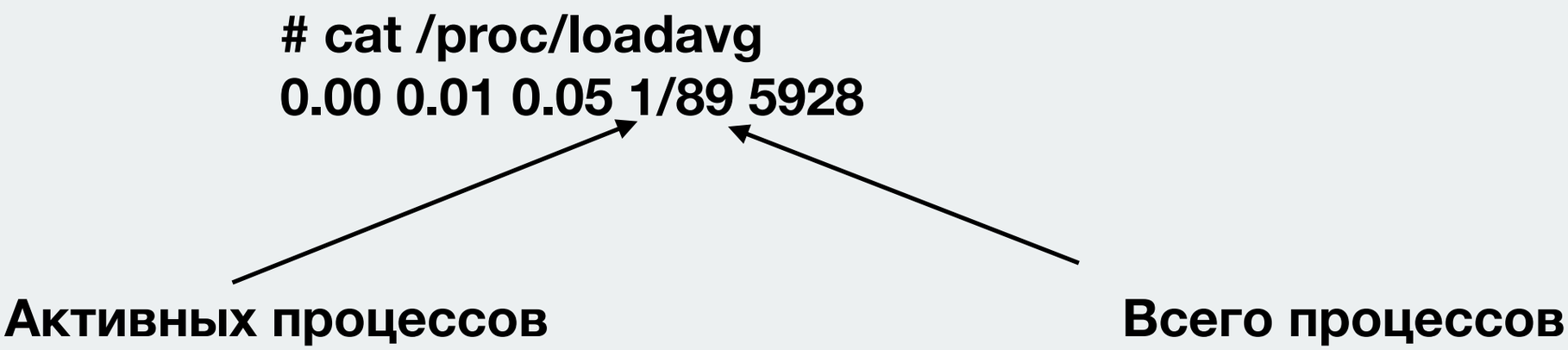
 # cat /proc/stat
cpu 12318 488 9095 86328268 47261 0 102 3946 0 0

man 5 proc

- **I/O wait**
- **idle task**
- **system mode**
- **user mode with low priority (nice)**
- **user mode**

CPU Usage

Борьба за процессорное время



Cpu usage маленький, LA большой, много процессов

CPU Usage

top

Disk Usage

CPU usage маленький, процессов мало, LA большой - мы упёрлись в IO

vmstat
iostat
iotop
cpu iowait (top)

Disk Usage

https://www.kernel.org/doc/Documentation/iostats.txt

```
# cat /proc/diskstats
   8       0 sda 14569 1 514694 42456 116506 58787 15152456 1062240 0 539238 1104652
```

#13

Время, проведенное в IO, мс

Мониторинг: Dx/Dt*100

https://www.kernel.org/doc/Documentation/iostats.txt

Disk Usage

CPU usage маленький, процессов мало, LA большой - мы упёрлись в IO

Но IO - это еще и сеть

Смотрим, что конкретно происходит с процессом
strace
/proc/<pid>/fd

Продолжаем про неотзывчивость

Всё неплохо, но клиенты не могут подключиться к сервису

Как обрабатываются соединения

Apache:
listen()
epoll()
accept()
fork()
<обработка>

Nginx:
listen()
fork()
epoll()
accept()
<обработка>

Открытые дескрипторы наследуются при fork()

Обрабатываем всплески

socket backlog

net.core.somaxconn = 128 → 1024

```
# ss -l
Recv-Q Send-Q
```

sysctl -w net.core.rmem_max=8388608
sysctl -w net.core.wmem_max=8388608
sysctl -w net.core.rmem_default=65536
sysctl -w net.core.wmem_default=65536
sysctl -w net.ipv4.tcp_rmem='4096 87380 8388608'
sysctl -w net.ipv4.tcp_wmem='4096 65536 8388608'
sysctl -w net.ipv4.tcp_mem='8388608 8388608 8388608'

Таймауты

Большие таймауты - мина замедленного действия. Всегда и везде стоит урезать таймауты до адекватных значений

- **connect timeout: 3 sec max**
- **write timeout: 3 sec max**
- **read timeout - зависит отприложения**

Swap - неизбежное зло

Многие процессы используют swap вполне "легально". В swap ядро превентивно "складывает" давно неиспользованые блоки. При нехватке памяти при переключении контекста память процесса выгружается на диск, что бы через полсекунды загрузить обратно.

Есть ряд решений для систем с ограниченной памятью

- **zswap**
- **zram**

Обе технологии "обменивают" дисковые операции на ресурс CPU

на production vm.swappiness не стоит ставить в 0, но стоит ставить в 1

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-memory-tunables

TCP_DEFER_ACCEPT

Ядро ядро передаёт управление в accept(), только когда появились данные
Экономим переключения контекста

apache:
AcceptFilter

nginx:
listen deferred

OTUS

Статистические утилиты

atop vs sar

strace

Утилита, показывающая, чем занимается процесс: какие системные вызовы дёргает процесс

`strace -p <pid>` - режим отображения в реальном времени
`strace -c -p <pid>` - режим с накоплением статистики
`strace -ff -p <pid>` - режим с отслеживаем форков

Ничего не помогает

```
echo 1 > /proc/sys/kernel/sysrq
echo b > /proc/sysrq-trigger
```

Спасибо за внимание!

Вопросы?