CYBERTEC
DATA SCIENCE & POSTGRESQL

Linux Control Groups für PostgreSQL

Laurenz Albe
www.cybertec-postgresql.com

Mein erster Kontakt

Erste Erfahrungen (1)

CYBERTEC
DATA SCIENCE & POSTGRESQL

Support request: „Datenbank startet nicht”

```
database system was interrupted
database system was not properly shut down;
                        automatic recovery in progress
invalid record length at 0/44715B0: wanted 24, got 0
redo is not required
startup process (8017) was terminated by signal 9: Killed
aborting startup due to startup process failure
database system is shut down
```

Erste Erfahrungen (2)

CYBERTEC
DATA SCIENCE & POSTGRESQL

Kernel log: es war der OOM Killer

```
Memory cgroup out of memory
Killed process 8017 (postgres)
oom_reaper: reaped process 8017 (postgres)
```

Aber RAM war großteils ungenutzt:

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:          78870       13998       54155       12224       10717        2063
Swap:          7935         138        7797
```

Erste Erfahrungen (3)

CYBERTEC
DATA SCIENCE & POSTGRESQL

- mit einiger Hilfe konnten Linux Control Groups als Ursache identifiziert werden
- Memory-Limits waren zu restriktiv konfiguriert
- Zeit für mich, etwas über Control Groups zu lernen!

→ daher dieser Vortrag

Kernel-Resourcen beschränken

Warum Resourcen beschränken?

CYBERTEC
DATA SCIENCE & POSTGRESQL

Gründe, die Kernel-Resourcen (Memory, I/O, CPU) für PostgreSQL zu beschränken:

- Mehrere PostgreSQL-Cluster auf einer Maschine (und man möchte nicht Docker verwenden)
- PostgreSQL läuft mit anderer Software auf dem selben Rechner

Die Datenbanken und Applikationen sollen einander nicht „aushungern" können.

Andere Methoden (1)

CYBERTEC
DATA SCIENCE & POSTGRESQL

PostgreSQL-Parameter: `shared_buffers`, `work_mem`, `maintenance_work_mem`, `max_connections`, `effective_io_concurrency`, `max_files_per_process`

Vorteil: bequem in `postgresql.conf` konfigurierbar

Nachteil: kein „hartes" Memory-Limit, CPU und I/O können nicht beschränkt werden

$\rightarrow$ nur teilweise brauchbar, vor allem I/O ist oft ein Problem

Andere Methoden (2)

CYBERTEC
DATA SCIENCE & POSTGRESQL

`ulimit`/`setrlimit`: setzt Limits pro Prozess

- ziemlich unbrauchbar, denn PostgreSQL besteht aus vielen Prozessen
- Überschreiten des CPU-Limits führt zum Tod des Prozesses

`nice`/`ionice`: Priorität im CPU- oder I/O-Scheduler setzen

- ermöglicht verschiedene Prioritäten für verschiede PostgreSQL-Cluster
- Gefahr, dass Prozesse mit niedriger Priorität „ausgehungert" werden

Linux Control Groups

Laurenz Albe
www.cybertec-postgresql.com

Geschichte

A Linux kernel feature that allows processes to be organized in a hierarchy of groups whose resource usage can be limited or monitored.

- Entwicklung seit 2006, seit 2008 im Linux Kernel (2.6.24)
- C-Library `libcg` seit 2008 in Entwicklung (noch immer Beta)
- 2016 Einführung von cgroups v2 mit einem neuen, vereinheitlichten Design (noch immer unvollständig)
  - noch immer ziemlich "work in progress"
  - ich rede im Folgenden nur von cgroups v1

Architektur

CYBERTEC
DATA SCIENCE & POSTGRESQL

- Resourcen sind in “Controllers” gruppiert: `cpu`, `memory`, `blkio`, `cpuset`, ...
- für jeden Controller wird ein „Pseudo-Filesystem” angehängt (meist unter `/sys/fs/cgroup`)
- cgroups-Hierarchie entspricht den Unterverzeichnissen
- jeder Parameter entspricht einer Datei (z.B. `cpuset.cpus` oder `memory.usage_in_bytes`)
- die Datei `tasks` enthält alle Prozesse, die zur cgroup gehören
- Kindprozesse „erben” die cgroup
- Administrationsrechte für eine cgroup können auch an normale Betriebssystembenutzer vergeben werden

Controller

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `memory`: Limits/Monitoring für RAM und Swap
- `cpu`/`cpuacct`: Limits/Monitoring für CPU
- `blkio`: Limits/Monitoring für I/O auf "block devices"
- `cpuset`: Limits/Monitoring für NUMA nodes
- `devices`: Zugriffsberechtigungen auf Devices
- `freezer`: Prozesse anhalten („einfrieren")
- `net_cls`: kennzeichnet Netzwerkpakete für den Linux traffic controller (`tc`)
- `hugetlb`: Limits/Monitoring für Huge Memory Pages
- `pids`: Limit für die Anzahl der Prozesse
- `perf_event`: Gruppierung für Profiling mit "perf"

Die wichtigsten Parameter für Limits

Parameter zur Memory-Kontrolle

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `memory.limit_in_bytes`: gesamtes RAM für alle Prozesse in der cgroup, inklusive Filesystem Cache
- `memory.memsw.limit_in_bytes`: gesamtes RAM + Swap für alle Prozesse
- `memory.oom_control`: bestimmt, was bei Überschreitung des Limits geschieht
  - 0 (default): der OOM-Killer beendet den Prozess
  - 1: der Prozess wird angehalten, bis Memory in der cgroup frei wird

Das ist unerfreulich. Es wäre schöner, wenn es eine Option gäbe, die zu einem "Out Of Memory"-Fehler führt.

Parameter zur I/O-Beschränkung

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `blkio.throttle.read_bps_device`, `blkio.throttle.write_bps_device`: beschränkt die Bytes, die pro Sekunde gelesen oder geschrieben werden können
- `blkio.throttle.read_iops_device`, `blkio.throttle.write_iops_device`: beschränkt die Anzahl der I/O-Operationen pro Sekunde

Diese Limits müssen in der folgenden Form geschrieben werden: *device_major*:*device_minor limit*, z.B. 8:0 10240
(Mehrere Zeilen für mehrere Devices)

Funktioniert perfekt mit PostgreSQL!

Parameters zur CPU-Beschränkung

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `cpu.cfs_period_us`: Dauer einer "Zeitscheibe" in Mikrosekunden
- `cpu.cfs_quota_us`: Limit für die Zeit in Mikrosekunden, die die cgroup innerhalb einer Zeitscheibe bekommt

Beachte: `cpu.cfs_quota_us` kann größer als `cpu.cfs_period_us` sein, dann kann die cgroup mehr als eine CPU verwenden

- `cpu.shares`: ein relatives Gewicht, nach dem die Resource zwischen allen cgroups auf der selben Ebene verteilt wird

Parameters für NUMA-Architektur

CYBERTEC
DATA SCIENCE & POSTGRESQL

Der zugehörige Controller heißt "`cpuset`".

NUMA = "non-uniform memory access"

- Um für Maschinen mit vielen Prozessoren schnellen Memory-Zugriff zu ermöglichen, werden Memory und CPUs in mehrere "Nodes" aufgeteilt.
- Zugriff von einem Prozessor auf „sein" Memory ist schneller als auf das Memory anderer Nodes.
- Anzeige der Nodes mit `numactl --hardware`

Parameter:

- `cpuset.mems`: welche Memory-Nodes darf die cgroup verwenden
- `cpuset.cpus`: welche CPUs darf die cgroup verwenden

Linux Control Groups verwalten

Laurenz Albe
www.cybertec-postgresql.com

Verwaltung im Filesystem

CYBERTEC
DATA SCIENCE & POSTGRESQL

Man kann Linux cgroups direkt über das Filesystem verwalten

- cgroup erzeugen mit `mkdir`, entfernen mit `rmdir`

```
mkdir /sys/fs/cgroup/memory/postgres
```

- Prozess in eine cgroup einfügen durch Hinzufügen der PID zum `tasks`-File
- Parameter durch Lesen und Schreiben der zugehörigen Dateien verwalten:

```
cat /sys/fs/cgroup/memory/postgres/memory.max_usage_in_bytes
echo '0,2-3' > /sys/fs/cgroup/cpuset/postgres/cpuset.cpus
```

Vorteil: keine zusätzliche Software erforderlich

Nachteil: man muss wissen, wo das Filesystem gemountet ist

Verwaltung mit `libcgroup`

CYBERTEC
DATA SCIENCE & POSTGRESQL

Man kann `libcgroup` (C API) und die zugehörigen Werkzeuge verwenden:

- `cgcreate` und `cgdelete` zum Erstellen und Löschen
- `cgexec` startet einen Prozess in einer cgroup
- `cgclassify` verschiebt einen Prozess in eine cgroup
- `cgget` und `cgset` lesen und schreiben Parameter

Vorteil: angenehm zu verwenden, gut in Shell-Scripts

Nachteil: von RedHat “deprecated”, unsichere Zukunft

Verwaltung mit `systemd`

CYBERTEC
DATA SCIENCE & POSTGRESQL

`systemd` bietet ein Interface für cgroup-Limits auf Prozessen:

- Wird über Parameter in der `[Service]`-Sektion eingestellt: `MemoryMax`, `MemorySwapMax`, `IOReadBandwidthMax`, `CPUQuota`, ...

Vorteile:

- Bequem für über `systemd` verwaltete Services
- RedHat sagt, das ist die Zukunft

Nachteile:

- nicht alles wird unterstützt (z.B. `cpuset`)
- ist irgendwo im Übergang von cgroups v1 und cgroups v2
- Parameter laufender Systeme können nicht geändert werden

Verwaltung über Konfigurationsdatei

CYBERTEC
DATA SCIENCE & POSTGRESQL

Konfigurationsdatei `/etc/cgconfig.conf` kann mit `cgconfigparser` eingelesen werden (händisch oder durch `systemd` beim Boot):

```
group db_cage {
    memory {
        memory.limit_in_bytes = 1G;
        memory.memsw.limit_in_bytes = 1G;
    }
    cpu {
        cpu.cfs_period_us = 1000000;
        cpu.cfs_quota_us = 250000;
    }
}
```

Vorteil: praktisch für den Boot-Prozess

Die Extension pg_cgroups

Warum pg_cgroups?

CYBERTEC
DATA SCIENCE & POSTGRESQL

Da Linux cgroups nützlich für PostgreSQL sind, wünsche ich mir eine einfache Möglichkeit für den DBA, sie zu verwalten.

Das ist die Idee hinter `pg_cgroups`. Das Modul bietet

- einfaches Laden mit `shared_preload_libraries`, erzeugt automatisch eine cgroup für den Cluster
- die PostgreSQL-Prozesse werden in die cgroup verschoben
- Resourcen-Limits werden über `postgresql.conf` eingestellt
- Änderung zur Laufzeit über `postgresql.conf` oder ALTER SYSTEM

pg_cgroups installieren

CYBERTEC
DATA SCIENCE & POSTGRESQL

- Installation mit `make` und `make install`
- „leere" cgroup `/postgres` anlegen
  (am besten mit `/etc/cgconfig.conf` beim Systemstart),
  die vom Betriebssystem-Benutzer `postgres` administriert werden kann
- `shared_preload_libraries = 'pg_cgroups'`
- PostgreSQL neu starten

pg_cgroups verwenden

CYBERTEC
DATA SCIENCE & POSTGRESQL

- Parameter in `postgresql.conf` oder mit `ALTER SYSTEM` einstellen.
  (Superuser erforderlich!)
- aktivieren durch `pg_ctl reload` oder `SELECT pg_reload_conf()`

Parameter von `pg_cgroups`

Memory-Parameter

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `pg_cgroups.memory_limit`: verfügbares RAM in MB
- `pg_cgroups.swap_limit`: verfügbarer Swap in MB

Im Unterschied zu den cgroup-Parametern beschränkt `swap_limit` *nicht* RAM + Swap, sondern nur Swap.

Es werden nicht nur `shared_buffers` und privates Memory beschränkt, sondern auch der für den Cluster verfügbare Filesystem-Cache!

- `pg_cgroups.oom_killer`: (Defaulteinstellung `on`) bestimmt, was bei Grenzüberschreitung passiert: OOM-Killer oder Prozess anhalten ("suspend"), bis Speicher frei wird

Memory-Parameter (Verwendung)

CYBERTEC
DATA SCIENCE & POSTGRESQL

Egal, wie `pg_cgroups.oom_killer` eingestellt ist, ist das Verhalten bei Grenzüberschreitung nicht ideal:

- OOM-Killer führt zu Crash Recovery
- Suspend führt wahrscheinlich zum „Einfrieren" des gesamten Clusters, wenn niemand Memory freigibt

Empfehlung: Memory „traditionell" mit `shared_buffers` und `work_mem` beschränken und die cgroup-Parameter nur zur Begrenzung des Filesystem-Caches verwenden.

Damit kann man verhindern, dass ein PostgreSQL-Cluster dem anderen den Filesystem-Cache „wegfrisst".

Block-I/O-Parameter

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `pg_cgroups.read_bps_limit` und `pg_cgroups.write_bps_limit`: beschränken Lesen und Schreiben in Bytes/Sekunde
- `pg_cgroups.read_iops_limit` und `pg_cgroups.write_iops_limit`: beschränken Lesen und Schreiben in Operationen/Sekunde

Die Werte haben die Form „*device_major*:*device_minor limit*"

Bei mehreren Devices die Einträge mit Komma trennen!

Diese Parameter funktionieren sehr gut.

CPU-Parameter

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `pg_cgroups.cpu_share`: wieviele Mikrosekunden CPU darf PostgreSQL in einer Zeitscheibe von 100000 Mikrosekunden verbrauchen

Der Wert kann 100000 übersteigen, wenn die Maschine mehr als einen Core hat.

Bei Überschreiten wird der Prozess „gedrosselt".

Funktioniert sehr gut!

NUMA-Parameter

CYBERTEC
DATA SCIENCE & POSTGRESQL

- `pg_cgroups.memory_nodes`: in welchen NUMA-Knoten darf PostgreSQL RAM verwenden

Diese Einstellung ist nur auf NUMA-Maschinen sinnvoll.

- `pg_cgroups.cpus`: welche Cores kann PostgreSQL verwenden

Dieser Parameter ist auch auf Multiprozessor-Maschinen ohne NUMA sinnvoll als Alternative zu `pg_cgroups.cpu_share`.

Die Werte beider Parameter sind eine Komma-separierte Liste von Bereichen, zum Beispiel `0,2,7-10`

Wenn man beide Parameter setzt, sollten die Cores zu den eingestellten Memory-Nodes gehören.

CYBERTEC
DATA SCIENCE & POSTGRESQL

Wo finde ich `pg_cgroups`?

- verfügbar unter https://github.com/cybertec-postgresql/pg_cgroups/
- freie Software unter der PostgreSQL-Lizenz

mögliche zukünftige Entwicklung

- an cgroups v2 anpassen, wenn die Entwicklung fertig ist

Fragen?