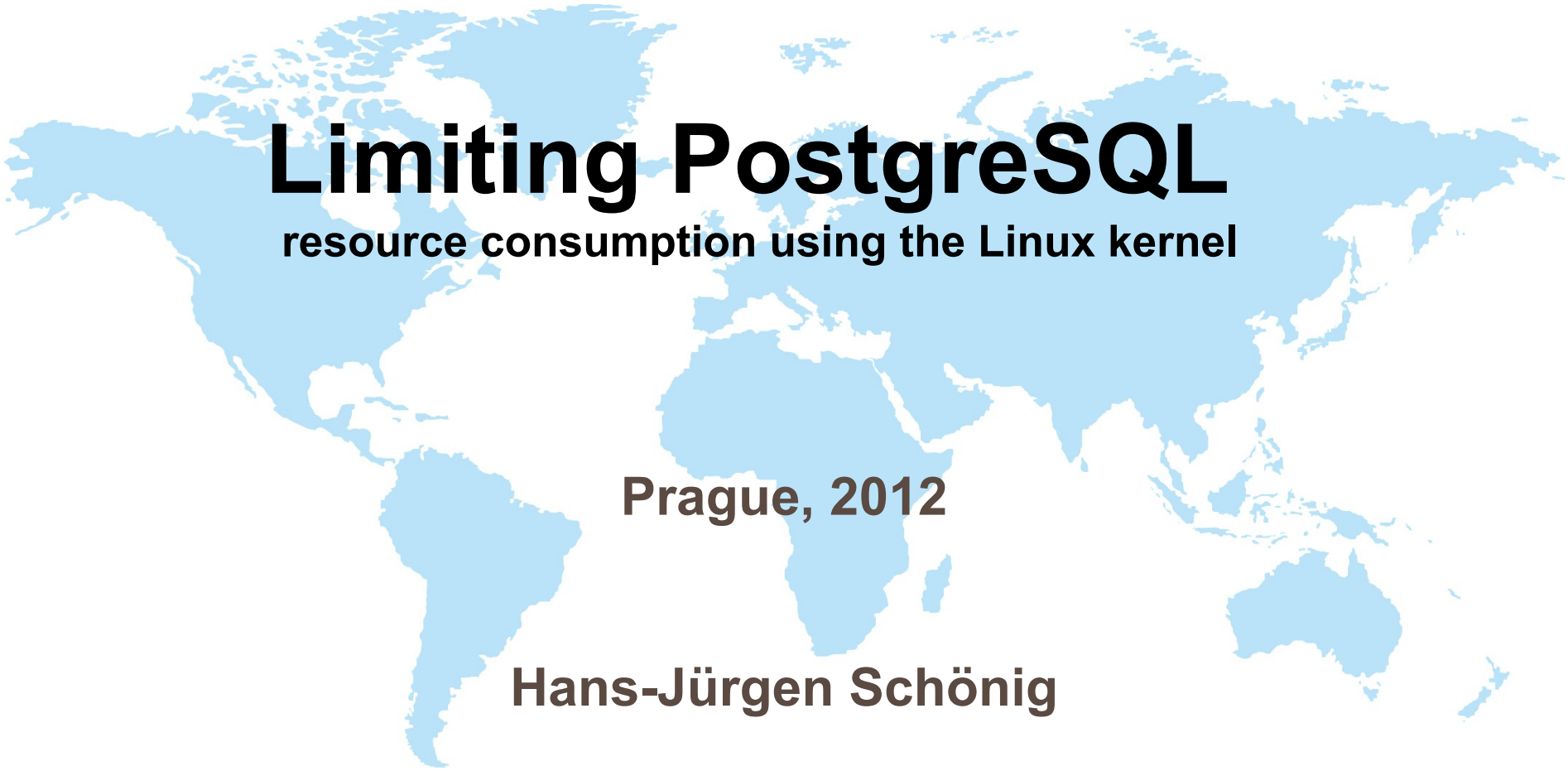


Limiting PostgreSQL

resource consumption using the Linux kernel

Prague, 2012

Hans-Jürgen Schönig

PostgreSQL: A typical setup

PostgreSQL optimized for ...

- maximum performance
- throughput

Parameters ...

- large memory settings
- long checkpoint intervals

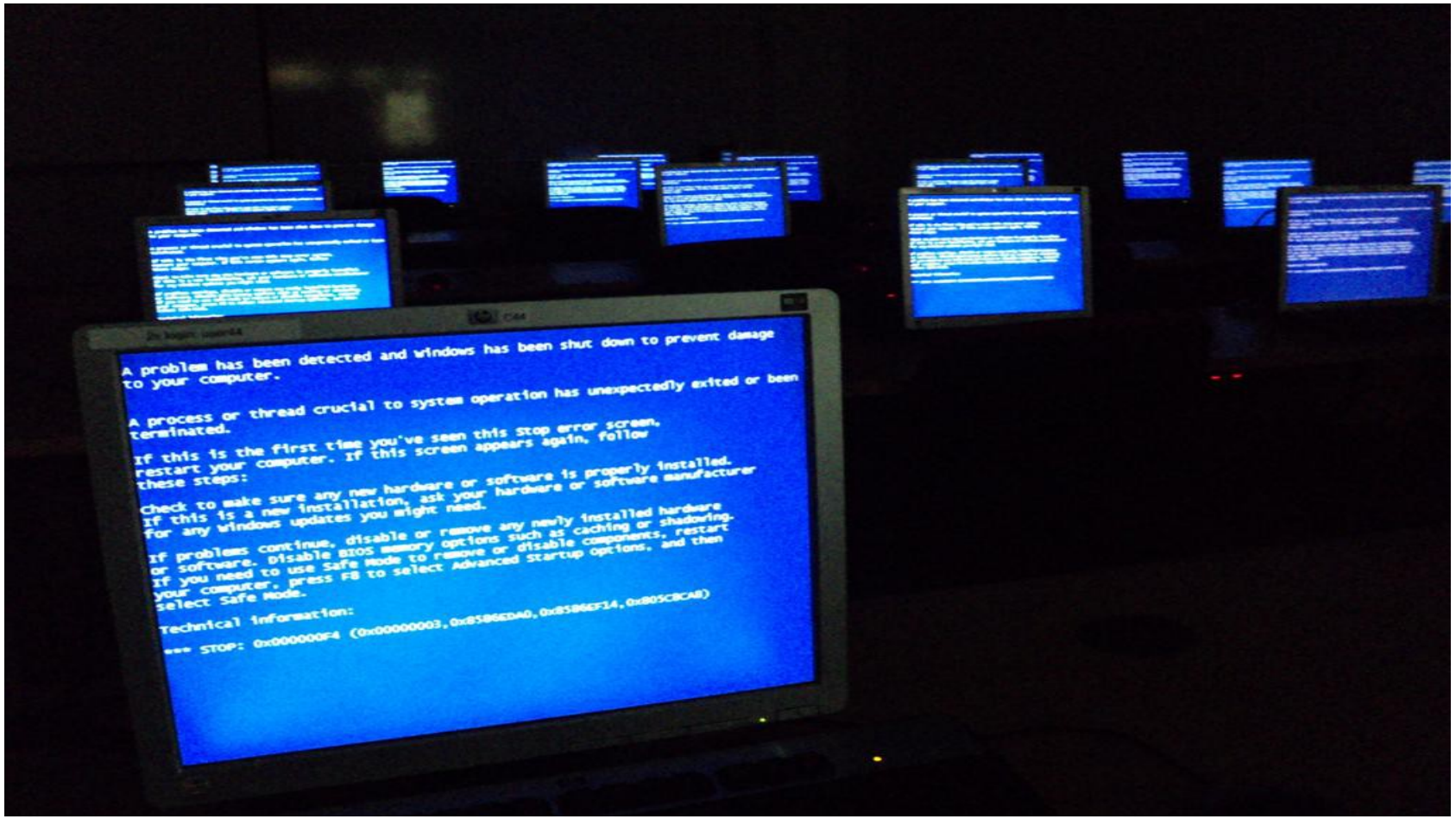
Coexistence

What if ...

- PostgreSQL is not alone on the system?
- RAM is limited and must be shared
- If not all CPUs can be used freely?
- Instances must be kept separate

What you want to avoid ...

CYBERTEC
The PostgreSQL Database Company



Want you want to make sure ...

One component MUST NOT KILL the entire system

Make sure things are isolated

NO side effects ...

<= This used
to be decoration

Using Linux onboard tools

Linux provides a mechanism called „cgroups“

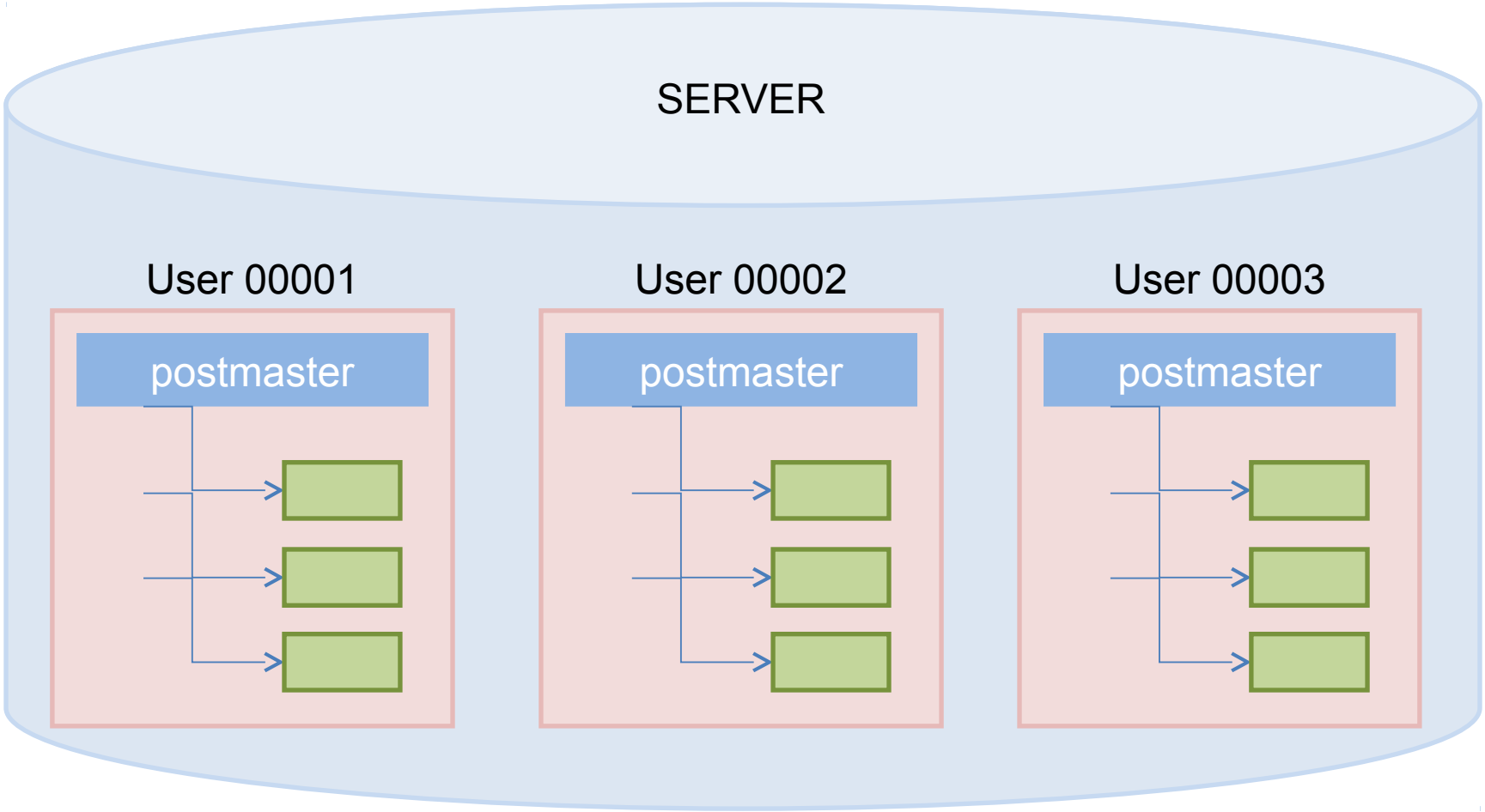
- Linux can isolate processes
- Capacities can be assigned to groups

=> CPUs („entire“ or „shares“)
=> Memory
=> I/O capacity

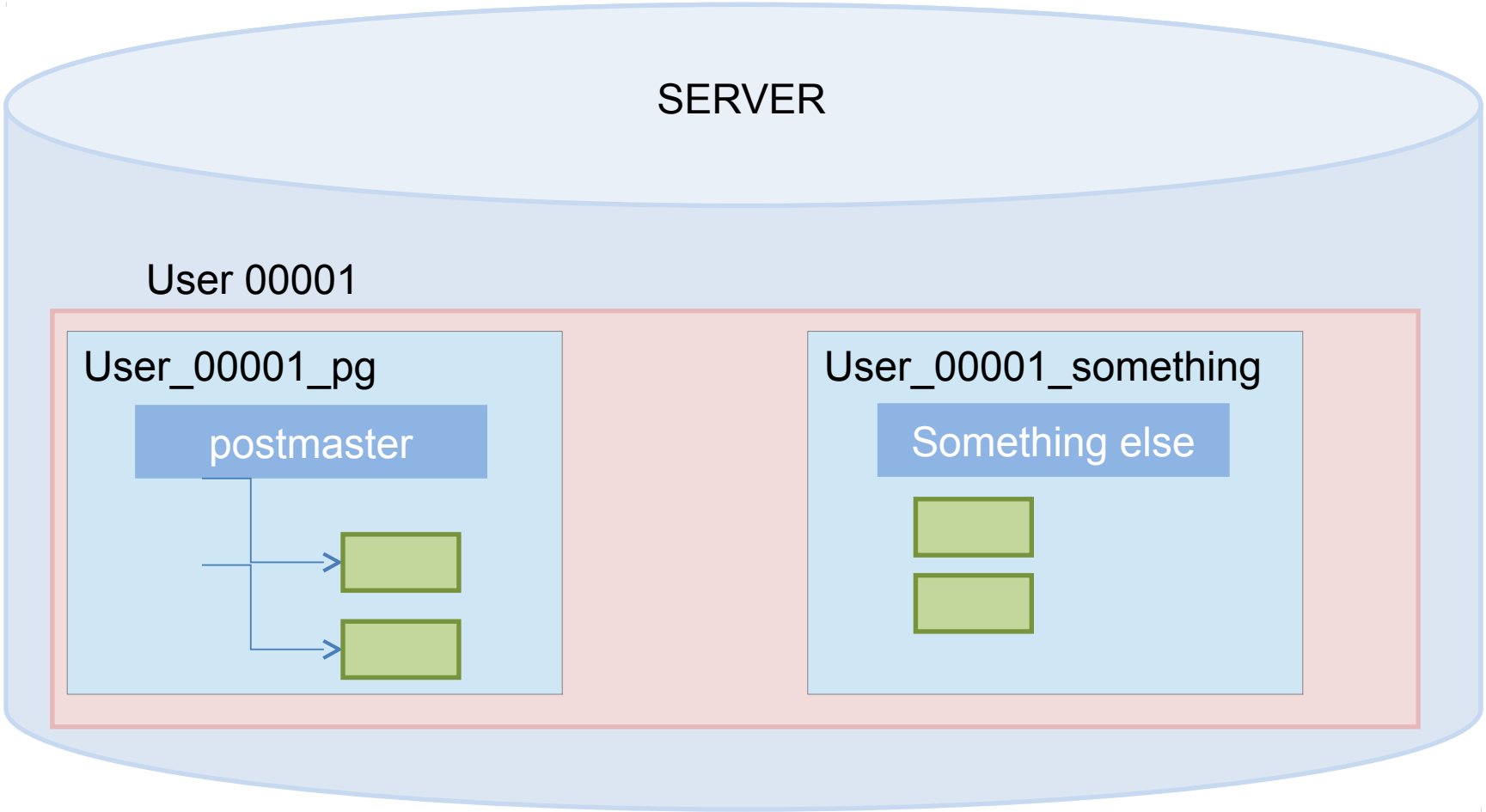
“cgroups” make sure that …

- Software A cannot kill software B by going nuts (eating too much RAM, etc.).

System architecture



Nested groups



Installing cgroups

Cgroups are a thing of the Linux kernel

Most distros provide a package to install the cgroups library + command line interface.

Installing on RHEL / Centos:

```
yum install –y libcgroup
```

How does it work? (1/5)

Various controllers are available:

- blkio: set limit for I/O devices
- cpu: scheduler for cgroup task access to CPU
- cpuacct: CPU reporting
- cpuset: assign CPUs and memory to a group
- devices: manage access to devices by task
- freezer: suspend tasks
- memory: limit memory
- net_cls: network limitations
- ns: namespace

How does it work? (2/5)

/etc/cgconfig.conf – configure groups

```
mount {
        cpu = /cgroup/main;
        cpuacct = /cgroup/main;
        memory = /cgroup/main;
        blkio = /cgroup/main;
}

group user000001 {
    perm {
        admin {
            uid = root;
            gid = root;
        }
        task {
            uid = user000001;
            gid = user000001;
        }
    }
    memory {
        memory.limit_in_bytes = 256M;
    }
    cpu {
        cpu.shares = 200;
    }
}
```

How does it work? (3/5)

Moving a process to a group:

CGCLASSIFY(1) libcgroup Manual CGCLASSIFY(1)

NAME
cgclassify - move running task(s) to given cgroups

SYNOPSIS
cgclassify [-g <controllers>:<path>] [--sticky | --cancel-sticky] <pidlist>

DESCRIPTION
this command moves processes defined by the list of
processes (pidlist) to given control groups.

The pids in the pidlist are separated by spaces

How does it work? (4/5)

Automatically move a process to a group /etc/cgrules.conf using cgred:

User	controller	destination
user000001	cpu,cpuacct,blkio,memory	/user000001
user000002	cpu,cpuacct,blkio,memory	/user000002
user000003	cpu,cpuacct,blkio,memory	/user000003

Cgroups can be nested

=> allows fine grained configuration

How does it work? (5/5)

- **Put different PostgreSQL instances into groups**
- **Make CPU / memory are assigned**
- **Processes must be in the right control group**
- **Cgred will make sure new connections will be moved to the correct group automatically**

  ⇒ no more work after setup

  ⇒ reliable

Reporting (1/2)

Just read those files ...

```
[root@paula cpuacct]# cat cpuacct.usage_percpu       <= we got 4 cores here
78324437172949 43026740916589 78819068266535 51258009493494

[root@paula cpuacct]# cat cpuacct.stat
user 10002466
system 4730954
```

NOTE:

Mind the CPU data; the unit depends on the kernel version.
Make sure you take this into account when using older kernels.

Use a function to convert to milliseconds / hours / etc.

Reporting (2/2)

Create nice reports for each DB instance / component

Theoretically you can track a single process

⇒ not reasonable in most cases

Writing an SQL wrapper to provide a custom policy would be pretty simple

Blkio details

blkio.io_merged blkio.io_queued blkio.io_service_bytes blkio.io_serviced blkio.io_service_time blkio.io_wait_time blkio.reset_stats blkio.sectors blkio.throttle.io_service_bytes blkio.throttle.io_serviced blkio.throttle.read_bps_device	blkio.throttle.read_iops_device blkio.throttle.write_bps_device blkio.throttle.write_iops_device blkio.time blkio.weight blkio.weight_device

CPU details

cgroup.procs
cpu.cfs_period_us
cpu.cfs_quota_us
cpu.rt_period_us
cpu.rt_runtime_us
cpu.shares
cpu.stat

CPU set details

cgroup.procs
cpuset.cpu_exclusive
cpuset.cpus <= e.g: “0-2,16”
cpuset.mem_exclusive
cpuset.mem_hardwall
cpuset.memory_migrate
cpuset.memory_pressure
cpuset.memory_pressure_enabled
cpuset.memory_spread_page
cpuset.memory_spread_slab
cpuset.mems
cpuset.sched_load_balance
cpuset.sched_relax_domain_level

Device details

cgroup.procs
devices.allow
devices.deny
devices.list

An example (/etc/cgconfig.conf)

```
group blkdev {
    devices {
        devices.deny="b 8:16 mrw"; # Deny access to /dev/sdb
        devices.deny="b 8:32 mrw"; # Deny access to /dev/sdc
        devices.deny="b 8:48 mrw"; # Deny access to /dev/sdd
    }
}
```

Memory details (1/2)

memory.failcnt
memory.force_empty
memory.limit_in_bytes
memory.max_usage_in_bytes
memory.memsw.failcnt
memory.memsw.limit_in_bytes
memory.memsw.max_usage_in_bytes
memory.memsw.usage_in_bytes
memory.move_charge_at_immigrate
memory.soft_limit_in_bytes
memory.stat
memory.swappiness
memory.usage_in_bytes
memory.use_hierarchy

Memory details (2/2)

[root@paula memory]# **cat memory.stat**
cache 6353096704
rss 90738688
mapped_file 114450432
pgpgin 4948404727
pgpgout 4946930148
swap 0
inactive_anon 91090944
active_anon 75034624
inactive_file 5703860224
active_file 573825024
unevictable 0
hierarchical_memory_limit 854775807
hierarchical_memsw_limit 854775807
total_cache 6353096704
total_rss 90738688
total_mapped_file 114450432
total_pgpgin 4948404727
total_pgpgout 4946930148
total_swap 0
total_inactive_anon 91090944
total_active_anon 75034624
total_inactive_file 5703860224
total_active_file 573825024
total_unevictable 0

Finally…

Any questions?

please feedback this talk
http://2012.postgresql.eu/feedback

Hans Jürgen Schönig

hs@cybertec.at | www.postgresql-support.de | www.cybertec.at