Delay accounting

Tasks encounter delays in execution when they wait for some kernel resource to become available e.g. a runnable task may wait for a free CPU to run on.

The per-task delay accounting functionality measures the delays experienced by a task while

a. waiting for a CPU (while being runnable)
b. completion of synchronous block I/O initiated by the task
c. swapping in pages
d. memory reclaim
e. thrashing page cache
f. direct compact

and makes these statistics available to userspace through the taskstats interface.

Such delays provide feedback for setting a task's cpu priority, io priority and rss limit values appropriately. Long delays for important tasks could be a trigger for raising its corresponding priority.

The functionality, through its use of the taskstats interface, also provides delay statistics aggregated for all tasks (or threads) belonging to a thread group (corresponding to a traditional Unix process). This is a commonly needed aggregation that is more efficiently done by the kernel.

Userspace utilities, particularly resource management applications, can also aggregate delay statistics into arbitrary groups. To enable this, delay statistics of a task are available both during its lifetime as well as on its exit, ensuring continuous and complete monitoring can be done.

Interface

Delay accounting uses the taskstats interface which is described in detail in a separate document in this directory. Taskstats returns a generic data structure to userspace corresponding to per-pid and per-tgid statistics. The delay accounting functionality populates specific fields of this structure. See

include/uapi/linux/taskstats.h

for a description of the fields pertaining to delay accounting. It will generally be in the form of counters returning the cumulative delay seen for cpu, sync block I/O, swapin, memory reclaim, thrash page cache, direct compact etc.

Taking the difference of two successive readings of a given counter (say cpu_delay_total) for a task will give the delay experienced by the task waiting for the corresponding resource in that interval.

When a task exits, records containing the per-task statistics are sent to userspace without requiring a command. If it is the last exiting task of a thread group, the per-tgid statistics are also sent. More details are given in the taskstats interface description.

The getdelays.c userspace utility in tools/accounting directory allows simple commands to be run and the corresponding delay statistics to be displayed. It also serves as an example of using the taskstats interface.

Usage

Compile the kernel with:

```
CONFIG_TASK_DELAY_ACCT=y
CONFIG_TASKSTATS=y
```

Delay accounting is disabled by default at boot up. To enable, add:

```
delayacct
```

to the kernel boot options. The rest of the instructions below assume this has been done. Alternatively, use sysctl kernel.task_delayacct to switch the state at runtime. Note however that only tasks started after enabling it will have delayacct information.

After the system has booted up, use a utility similar to getdelays.c to access the delays seen by a given task or a task group (tgid). The utility also allows a given command to be executed and the corresponding delays to be seen.

General format of the getdelays command:

```
getdelays [-dilv] [-t tgid] [-p pid]
```

Get delays, since system boot, for pid 10:

```
# ./getdelays -d -p 10
(output similar to next case)
```

Get sum of delays, since system boot, for all pids with tgid 5:

```
# ./getdelays -d -t 5
print delayacct stats ON
TGID    5


CPU             count     real total  virtual total    delay total  delay average
                    8        7000000        6872122        3382277          0.423ms
IO              count    delay total  delay average
                    0              0              0ms
SWAP            count    delay total  delay average
                    0              0              0ms
RECLAIM         count    delay total  delay average
                    0              0              0ms
THRASHING       count    delay total  delay average
                    0              0              0ms
COMPACT         count    delay total  delay average
                    0              0              0ms
```

Get IO accounting for pid 1, it works only with -p:

```
# ./getdelays -i -p 1
printing IO accounting
linuxrc: read=65536, write=0, cancelled_write=0
```

The above command can be used with -v to get more debug information.