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

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/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 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 this 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 enabled by default at boot up.
To disable, add
    nodelayacct
to the kernel boot options. The rest of the instructions
below assume this has not been done.

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 [-t tgid] [-p pid] [-c cmd...]

Get delays, since system boot, for pid 10
./getdelays -p 10
(output similar to next case)

Get sum of delays, since system boot, for all pids with tgid 5
./getdelays -t 5

CPU	count	real total	virtual total	delay total
	7876	92005750	100000000	24001500
IO	count	delay total		
	0	0		
SWAP	count	delay total		
	0	0		
RECLAIM	count	delay total		
	0	0		

Get delays seen in executing a given simple command
./getdelays -c ls /

```
bin    data1  data3  data5  dev  home  media  opt   root  srv        sys  usr
boot   data2  data4  data6  etc  lib   mnt    proc  sbin  subdomain  tmp  var
```


CPU	count	real total	virtual total	delay total
	6	4000250	4000000	0
IO	count	delay total		

	count	delay total
	0	0
SWAP	count	delay total
	0	0
RECLAIM	count	delay total
	0	0