

Fix pacct bug in multithreading case.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg09551.html>

- *From:* KaiGai Kohei <kaigai@xxxxxxxxxxxxx>
 - *Date:* Tue, 28 Mar 2006 20:43:49 +0900
-

Hello,

I noticed a bug on the process accounting facility.
In multi-threading process, some data would be recorded incorrectly when the group_leader dies earlier than one or more threads.
The attached patch fixes this problem.

See below. 'bugacct' is a test program that create a worker thread after 4 seconds sleeping, then the group_leader dies soon.
The worker thread consume CPU/Memory for 6 seconds, then exit.
We can estimate 10 seconds as etime and 6 seconds as stime + utime.
This is a sample program which the group_leader dies earlier than other threads.

The results of same binary execution on different kernel are below.

```
--- accounted records -----  
| btime | utime | stime | etime | minflt | majflt | comm |  
original | 13:16:40 | 0.00 | 0.00 | 6.10 | 171 | 0 | bugacct |  
patched | 13:20:21 | 5.83 | 0.18 | 10.03 | 32776 | 0 | bugacct |  
(* ) bugacct allocates 128MB memory, thus 128MB / 4KB = 32768 of minflt is  
appropriate.
```

```
--- Test results in original kernel -----  
$ date; time -p ./bugacct  
Tue Mar 28 13:16:36 JST 2006 <- But pacct said btime is 13:16:40  
real 10.11 <- But pacct said etime is 6.10  
user 5.96 <- But pacct said utime is 0.00  
sys 0.14 <- But pacct said stime is 0.00  
$
```

```
--- Test results in patched kernel -----  
$ date; time -p ./bugacct  
Tue Mar 28 13:20:21 JST 2006  
real 10.04  
user 5.83  
sys 0.19  
$
```

In the original 2.6.16 kernel, pacct records btime, utime, stime, etime and minflt incorrectly. In my opinion, this problem is caused by an assumption

Fix pacct bug in multithreading case.

that group_leader dies last.

The following section calculates process running time for etime and btime.

But it means running time of the thread that dies last, not process.

The start_time of the first thread in the process (group_leader) should be reduced from uptime to calculate etime and btime correctly.

```
----- do_acct_process() in kernel/acct.c:
/* calculate run_time in nsec*/
do_posix_clock_monotonic_gettime(&uptime);
run_time = (u64)uptime.tv_sec*NSEC_PER_SEC + uptime.tv_nsec;
run_time -= (u64)current->start_time.tv_sec*NSEC_PER_SEC
+ current->start_time.tv_nsec;
-----
```

The following section calculates stime and utime of the process.

But it might count the utime and stime of the group_leader duplicatly and ignore the utime and stime of the thread dies last, when one or more threads remain after group_leader dead.

The ac_utime should be calculated as the sum of the signal->utime and utime of the thread dies last. The ac_stime should be done also.

```
----- do_acct_process() in kernel/acct.c:
jiffies = cputime_to_jiffies(cputime_add(current->group_leader->utime,
current->signal->utime));
ac.ac_utime = encode_comp_t(jiffies_to_AHZ(jiffies));
jiffies = cputime_to_jiffies(cputime_add(current->group_leader->stime,
current->signal->stime));
ac.ac_stime = encode_comp_t(jiffies_to_AHZ(jiffies));
-----
```

The part of the minflt/majflt calculation has same problem.

This patch solves those problems, I think.

Any comments are welcome. Thanks.

--

Linux Promotion Center, NEC

KaiGai Kohei <kaigai@xxxxxxxxxxxxxx>

--- linux-2.6.16/kernel/acct.c 2006-03-20 14:53:29.000000000 +0900

+++ linux-2.6.16-kg/kernel/acct.c 2006-03-27 16:25:11.000000000 +0900

@@ -449,8 +449,8 @@ static void do_acct_process(long exitcod

```
/* calculate run_time in nsec*/
```

```
do_posix_clock_monotonic_gettime(&uptime);
```

```
run_time = (u64)uptime.tv_sec*NSEC_PER_SEC + uptime.tv_nsec;
```

```
- run_time -= (u64)current->start_time.tv_sec*NSEC_PER_SEC
```

```
- + current->start_time.tv_nsec;
```

```
+ run_time -= (u64)current->group_leader->start_time.tv_sec * NSEC_PER_SEC
```

```
+ + current->group_leader->start_time.tv_nsec;
```

```
/* convert nsec -> AHZ */
```

```
elapsed = nsec_to_AHZ(run_time);
```

```
#if ACCT_VERSION==3
```

```
@@ -469,10 +469,10 @@ static void do_acct_process(long exitcod
```

```
#endif
```

Fix pacct bug in multithreading case.

Fix pacct bug in multithreading case.

```
do_div(elapsed, AHZ);
ac.ac_btime = xtime.tv_sec - elapsed;
- jiffies = cputime_to_jiffies(cputime_add(current->group_leader->utime,
+ jiffies = cputime_to_jiffies(cputime_add(current->utime,
current->signal->utime));
ac.ac_utime = encode_comp_t(jiffies_to_AHZ(jiffies));
- jiffies = cputime_to_jiffies(cputime_add(current->group_leader->stime,
+ jiffies = cputime_to_jiffies(cputime_add(current->stime,
current->signal->stime));
ac.ac_stime = encode_comp_t(jiffies_to_AHZ(jiffies));
/* we really need to bite the bullet and change layout */
@@ -522,9 +522,9 @@ static void do_acct_process(long exitcod
ac.ac_io = encode_comp_t(0 /* current->io_usage */); /* %% */
ac.ac_rw = encode_comp_t(ac.ac_io / 1024);
ac.ac_minflt = encode_comp_t(current->signal->min_flt +
- current->group_leader->min_flt);
+ current->min_flt);
ac.ac_majflt = encode_comp_t(current->signal->maj_flt +
- current->group_leader->maj_flt);
+ current->maj_flt);
ac.ac_swaps = encode_comp_t(0);
ac.ac_exitcode = exitcode;
```

```
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/time.h>
```

```
#define BUFFER_SIZE (128 * 1024 * 1024) /* 128MB */
```

```
void *mychild(void *buffer) {
struct timeval tv;
u_int64_t t1, t2;

gettimeofday(&tv, NULL);
t1 = tv.tv_sec * 1000000 + tv.tv_usec;
t2 = t1 + 6 * 1000000;
/* heavy CPU/Mem job */
srand(tv.tv_usec);
while(t1 < t2) {
memset(buffer, rand(), BUFFER_SIZE);
gettimeofday(&tv, NULL);
t1 = tv.tv_sec * 1000000 + tv.tv_usec;
}
return NULL;
}
```

```
int main(int argc, char *argv[]) {
pthread_t pthread;
void *buffer = NULL;
```

Fix pacct bug in multithreading case.

```
buffer = malloc(BUFFER_SIZE);
if (!buffer)
return 1;

sleep(4);
pthread_create(&pthread, NULL, mychild, buffer);
sleep(1);
pthread_exit(0);
}
```