

[PATCH] sched: smpnice work around for active_load_balance()

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg09482.html>

- *From:* Peter Williams <pwil3058@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 28 Mar 2006 17:00:50 +1100
-

Problem:

It is undesirable for HT/MC packages to have more than one of their CPUs busy if there are other packages that have all of their CPUs idle. This involves moving the only running task (i.e. the one actually on the CPU) off on to another CPU and is achieved by using `active_load_balance()` and relying on the fact that (when it starts) the queue's migration thread will preempt the sole running task and (therefore) make it movable. The migration thread then moves it to an idle package.

Unfortunately, the mechanism for setting the run queues `active_balance` flag is buried deep inside `load_balance()` and relies heavily on `find_busiest_group()` and `find_busiest_queue()` reporting success even if the busiest queue has only one task running. This requirement is not currently met.

Solution:

The long term solution to this problem is provide an alternative mechanism for triggering active load balancing for run queues that need it.

However, in the meantime, this patch modifies `find_busiest_group()` so that (when `idle != NEWLY_IDLE`) it prefers groups with at least one CPU with more than one running task over those with only one and modifies `find_busiest_queue()` so that (when `idle != NEWLY_IDLE`) it prefers queues with more than one running task over those with only one. This means that the "busiest" queue found in `load_balance()` will only have one or less runnable tasks if there were no queues with more than one runnable task. When called in `load_balance_newidle()` they will have the existing functionality. These measure will prevent high load weight tasks that have a CPU to themselves from suppressing load balancing between other queues.

NB This patch should be backed out when a proper fix for the problems inherit in HT and MC systems is implemented.

Signed-off-by: Peter Williams <pwil3058@xxxxxxxxxxxxxxxx>

Peter

Peter Williams pwil3058@xxxxxxxxxxxxxxxx

[PATCH] sched: smpnprc work around for active_load_balance()

"Learning, n. The kind of ignorance distinguishing the studious."
— Ambrose Bierce

Index: MM-2.6.X/kernel/sched.c

----- MM-2.6.X.orig/kernel/sched.c 2006-03-27 17:02:53.000000000 +1100

+++ MM-2.6.X/kernel/sched.c 2006-03-28 16:57:58.000000000 +1100

@@ -2098,6 +2098,7 @@ find_busiest_group(struct sched_domain *
unsigned long max_pull;
unsigned long busiest_load_per_task, busiest_nr_running;
unsigned long this_load_per_task, this_nr_running;
+ unsigned int busiest_has_loaded_cpus = idle == NEWLY_IDLE;
int load_idx;

max_load = this_load = total_load = total_pwr = 0;
@@ -2152,7 +2153,15 @@ find_busiest_group(struct sched_domain *
this = group;
this_nr_running = sum_nr_running;
this_load_per_task = sum_weighted_load;
- } else if (avg_load > max_load && nr_loaded_cpus) {
+ } else if (nr_loaded_cpus) {
+ if (avg_load > max_load || !busiest_has_loaded_cpus) {
+ max_load = avg_load;
+ busiest = group;
+ busiest_nr_running = sum_nr_running;
+ busiest_load_per_task = sum_weighted_load;
+ busiest_has_loaded_cpus = 1;
+ }
+ } else if (!busiest_has_loaded_cpus && avg_load < max_load) {
max_load = avg_load;
busiest = group;
busiest_nr_running = sum_nr_running;
@@ -2161,7 +2170,7 @@ find_busiest_group(struct sched_domain *
group = group->next;
} while (group != sd->groups);

- if (!busiest || this_load >= max_load)
+ if (!busiest || this_load >= max_load || busiest_nr_running == 0)
goto out_balanced;

avg_load = (SCHED_LOAD_SCALE * total_load) / total_pwr;
@@ -2265,12 +2274,19 @@ static runqueue_t *find_busiest_queue(st
{
unsigned long max_load = 0;
runqueue_t *busiest = NULL, *rqi;
+ unsigned int busiest_is_loaded = idle == NEWLY_IDLE;
int i;

for_each_cpu_mask(i, group->cpumask) {
rqi = cpu_rq(i);

[PATCH] sched: smpnprc work around for active_load_balance()

[PATCH] sched: smpnice work around for active_load_balance()

```
- if (rqi->raw_weighted_load > max_load && rqi->nr_running > 1) {  
+ if (rqi->nr_running > 1) {  
+ if (rqi->raw_weighted_load > max_load || !busiest_is_loaded) {  
+ max_load = rqi->raw_weighted_load;  
+ busiest = rqi;  
+ busiest_is_loaded = 1;  
+ }  
+ } else if (!busiest_is_loaded && rqi->raw_weighted_load > max_load) {  
max_load = rqi->raw_weighted_load;  
busiest = rqi;  
}
```