Setting Processor Affinity on Linux

Alex Podelko
Oracle

I needed to check if setting processor affinity would allow me to minimize mutual performance impact of server programs running inside a Linux VM. Setting processor affinity in Linux with taskset looks pretty straightforward:

```
taskset -c -p <cpu#> <pid>
```

However, when I ran a test, I noticed that the programs still used all CPUs. So I started an investigation. Truly speaking, I suspected that it might be a virtualization effect – but no. I discovered that, to set processor affinity for a multithreaded application (such as the one I worked with), I need to set affinity for all tasks in **/proc/<pid>/task**. This can be done with something like:

```
for p in `ls /proc/<pid>/task`; do taskset -p -c <cpu#>  $p; done
```

This deals with "threads are essentially just processes with a shared address space on Linux".

As soon as I ran the code above for the pid, the process was limited to one processor and affinity worked fine.

So the problem was that I had set affinity only for the parent process, but not for the all tasks in **/proc/<pid>/task**. Child processes inherit affinity when they are started. But in this case the child processes all were started before I set affinity for the main process – so the child processes did not automatically pick up the change to the parent process.

The process may be started with affinity from the beginning. The following, for example, works fine too:

```
taskset -c <cpu#>  <start_command>
```

But if you need to set affinity for already running processes, don't forget to set it for all tasks in **/proc/<pid>/task** too.