Linux Plumbers Conference 2018

Contribution ID: **206** Type: **not specified**

Scheduler and pipe sleep wakeup scalability

Wednesday 14 November 2018 10:00 (30 minutes)

1) Scalability of scheduler idle cpu and core search on systems with large number of cpus

Current select_idle_sibling first tries to find a fully idle core using select_idle_core which can potentially search all cores and if it fails it finds any idle cpu using select_idle_cpu. select_idle_cpu can potentially search all cpus in the llc domain. These don't scale for large llc domains and will only get worse with more cores in future. Spending too much time in the scheduler will hurt performance of very context switch intensive workloads. A more scalable way to do the search is desirable which is *not* O(no. of cpus) or O(no. of cores) in worst case.

2) Scalability of idle cpu stealing on systems with large number of cpus and domains

When a cpu becomes idles it tries to steal threads from other overloaded cpus using idle_balance. idle_balance does more work because it searches widely for the busiest CPU to offload, so to limit its CPU consumption, it declines to search if the system is too busy. A more scalable/lightweight way of stealing is desirable so that we can always try to steal with very little cost.

3) Discuss workloads that use pipes and can benefit from pipe busy waits

When pipe is full or empty a thread goes to sleep immediately. If the sleep wakeup happens very fast, the cost of sleep wakeup overhead can hurt a very context switch sensitive workload that is using pipes heavily. A few microseconds of busy wait before sleeping can avoid the overhead and improve the performance. Network sockets has similar capability. So far hackbench with pipe shows huge improvements, want to discuss other potential use cases.

Presenter: MAZUMDAR, Subhra

Session Classification: Performance and Scalability MC