Paper Critique – Week 3

Name: PHAM CONG VINH Student ID: 2019711010

The Linux Scheduler: a Decade of Wasted Cores

The subject of this paper is about the Linux's scheduler. Because the evolution of hardware technology which Linux's scheduler was being improved, optimized in order to take advantage of, but it also introduced bugs that broke the scheduler maintaining invariant: make sure that ready threads are scheduled on available cores. This made some CPU core being idle even there are ready threads (task) in waiting queues.

The authors of this paper discovered 4 bugs which are: The Group Imbalance bug, The Scheduling Group Construction bug, The Overload-on-Wakeup bug and The Missing Scheduling Domains bug. The symptom of this kind of bug is hardly detected by conventional testing techniques and debugging tools because it didn't make the system crashing or halting but stealthy eating away CPU performance for second. The visualization tools was developed in order to detecting these kinds of bug. Moreover, authors proposed a promised architecture (basic core and modules) for Linux scheduler.

Most of the performance evaluation testing was done with server, database workload oriented application. Additional test or discussion how Linux's scheduler patched by authors enhance desktop or interactive application (multimedia, GUI…) performance will be much better.

Arachne: Core-Aware Thread Management

The subject of this paper is about a proposed thread management library named Arachne. Usually, operating system virtualize resources (specifically, in this case is CPU core) to thread. This made application difficult to adjust it' internal parallelism to match resources allocated to it. In other word, application can't use their application-specific knowledge in order to optimize resource utilization. By employing Arachne, application can effective manage CPU core utilization at user level without changing kernel, scheduler.

Arachne library contained 3 main components: Arachne Runtime, Arache Core Arbiter and Core Polices. The authors tested their proposed solution by pratical application (memcached, RAMCloud) and compare with other solution (Go, uThread, std::thread). This is persuasive and helped strengthen their solution. Also, author's idea about how cluster schedulers for datacenter-scale applications will be beneficial by employing their solution is interested and promised.

In the paper, authors stated that their solution will enhance application that required both high throughput and low latency in term of design each connection (user) as a short-live thread. But there are some research about Event-driven paradigm design stated that design connection as thread isn't reasonable. A discussion about this situation at Related Work Section will be much better.