ECE5658: Operating Systems Design

Paper Reading Assignment

2019712410 Jeon Hyung-Joon

Arachne: Core-Aware Thread Management

Summary:

The objective of this paper is to present the Arachne architecture. It provides both low latency and high throughput for applications with extremely short-lived threads. Arachne is core-aware; i.e. thread management is done at application-level. An application can determine the number of cores it needs from its own load. It is able to accurately perceive allocation status of CPU cores, and it can enable its thread to be run upon the cores chosen for use. Such job is guided by the so-called core arbiter module, which assigns available cores to application threads.

Strengths and Weaknesses:

The idea of introducing user-level thread management seems promising. The threading primitives in Arachne are clearly shown to be efficient compared to other threading frameworks. From the core-aware approach, low-latency applications can have huge benefit from using the core arbiter and the runtime. Arachne's internal mechanisms, such as its queue-less approach to thread scheduling and its mechanisms for core estimation and core allocation, contribute to performance. Such architecture might seem appropriate for use in enabling applications to access datacenter storages for low-latency remote data management.

However, there are couple of weaknesses that need to be considered. The scale of testing environment seems to be limited. The tests are done only on 8-core environment, which is relatively small. Memory overhead, in addition, must be checked for verifying stability of the Arachne architecture.

Suggestions for Improvement:

I suggest improving this paper by performing further experiments on bigger-scale environments. Moreover, experiments on the existence of memory overhead can be performed to prove that the Arachne system has little issues in terms of memory consumption.