CS423 Summary: Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism

Hongpeng Guo

February 16, 2019

Area:

This paper pointed out that the kernel level threading is not a good level of abstraction for multiprocessor systems today. The authors designed a hybrid threading idea for both efficient and low restriction threading. Overall, this paper lies in the area of multi-thread scheduling.

Problem:

At the time of this paper, user level threading and kernel level threading are two major principles for multiprocessor system scheduling. User level threading is efficient but object to a set of operation restrictions. Kernel level threading introduces much more overhead but is more flexible. It is a critical problem to design of system scheduling scheme which is fast and flexible.

Methodology and Solutions:

The authors modified the kernel level threading by adding interfaces in kernel level for the use of user level. During the execution process, the critical sections will be duplicated to user level. Virtual memory space and virtual processors are also introduced in this design.

Results:

- The cost of user-level thread operations in this paper is essentially the same as those of the multiprocessor systems adopting user level threading, such as FastThreads.
- The Upcall performance is significantly worse than existing systems, such as Topaz.
- Testing on application programs such as N-body problem. This paper illustrated a similar speedup as FastThread and less overall execution time than FastThread and Topaz.

Takeaway:

Sometimes it is essential to break the traditional system model, such as the threading abstraction in this paper. A kernel interface and user level management together may achieve schedule performance than traditional abstractions.