System Design for Chip Multiprocessor

Bin Ren *
Computer Laboratory
University of Cambridge
bin.ren@cl.cam.ac.uk
www.cl.cam.ac.uk/~br260

1 Problem

Recent years have witnessed a shift of focus to exploiting TLP by integrating multiple hardware cores/threads into one silicon chip. In CMP/MT processors, the interrelationships of logical processors are no longer uniform. Logical processors in the same core share a lower level cache, whilst those in different cores share a higher level cache. This nonuniformity poses new problems for an OS scheduler:

- Lock Compatibility

  In SMP systems, conventional locks generate a lot of memory bus transactions due to cache coherency protocols, resulting in very poor scalability. On CMP/MT processors where logical processors share caches, two processes can contend for locks entirely in caches. An OS scheduler that is fully aware of CMP/MT and *inter-process lock compatibility* can significantly reduce main memory accesses.

- Resource Compatibility

  On CMP/MT processors, processes that have compatible or even shared worksets should be scheduled on logical processors that share a lower level cache; processes that have incompatible or even conflicting worksets should be scheduled on logical processors that share a higher level cache. An OS scheduler that is fully aware of CMP/MT and *inter-process resource compatibility* can significantly reduce cache misses.

In summary, we need a system architecture that suits the nonuniform architecture of CMP/MT processors, defines proper metrics for inter-process lock and resource compatibility, employs effective methods to calculate the metrics, and uses the metrics to make optimal scheduling decisions.

*Funded by Intel Research, Cambridge

2 Solution

I propose *flow network architecture (FNA)*, a new system architecture that has two major facets: the structure of user-level applications and the algorithm of OS-level schedulers.

A FNA application consists of a network of *stages* representing the functional units of processing. Each stage has an *input event queue*, an *flow scheduler* and an *event handler*. The event handler is driven by threads. The arrival of a request generates an *event*. The event travels along a path of stages, forming a *flow*. As the event moves along this flow, the corresponding request is processed. Upon finish, a response is delivered and the event ceases to exist.

The major task of a FNA OS scheduler is to move as many events as possible through the network of stages (i.e. high throughput) and as fast as possible (i.e. low latency). In this way, the scheduling policy is transformed into a classic *flow network problem*. There have existed many algorithms to solve different flow network problems, such as *shortest path problem*, *maximum flow problem* and *minimum cost flow problem*.

Queues in FNA applications are equivalent to edges in flow networks. Edge capacity can be defined as the threshold of input event queues, which is mainly used for *load shedding*. The edge cost is defined as the *intra-stage cost* which is inherently related to the processing logic of the stage. The *inter-stage cost* is the cost of switching the execution from one stage to another, involving cache misses, TLB misses and pipeline flushes. The higher resource compatibility two stages have, the lower their inter-stage cost is. When two stages have high locking compatibility, it's desirable to schedule them one after another so that locking can be done in the cache without main memory accesses. Therefore, when one stage is about to be scheduled off CPU, stages that have high lock compatibility should be given a *locking bonus*.

At runtime, statistics such as cache misses and CPU idle time can be measured by performance counters. The major research focus is how to calculate *resource compatibility* and *lock compatibility* from these statistics, and later use them in flow network algorithms to make scheduling decisions.

3 Justification

FNA deals with the nonuniformity in CMP/MT processors by incorporating inter-stage resource and lock compatibility into scheduling decisions. FNA aims to reduce resource and lock contention while providing high concurrency for demanding server applications so that TLP can be exploited more efficiently and effectively. FNA also facilitates *online reconfiguration* and *fault tolerance*.