Heterogeneous Multicore Scheduling

Naman Jain and Prashanth Suresh

ECE 18-743 Poster Session
December 2014

1. Goals

Create a model of Performance v/s Power and strike a "sweet spot" to

- Satisfy power budget
- Performance requirements
- Die area constraints

❖ Key Aspects:

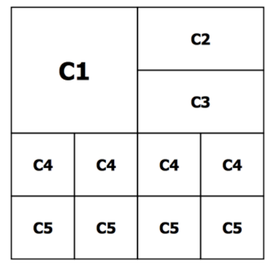
- Number of cores of each type
- Core configurations
- Optimal scheduling policy

❖ Contribution:

- A new performance model is designed for multi-threaded applications from the SPLASH-2 and PARSEC-2.1 benchmark suites.
- Scheduled applications on heterogeneous CMPs across a wide range of chip sizes and number of chips.

2. System Overview

- Execute parallel part on small cores for high throughput and low power
- Accelerate serialized sections using the large core



Solution: Heterogeneous multicore

Core Parameter	Big	Small
Dispatch Width	4	2
L1-I/D Cache size	64 KB	32 KB
L2 Cache (private)	256 KB	128 KB
Frequency	2.66 GHz	1.5 GHz
ROB Window	128	32

❖ Tools Used

- Performance modeling: *Sniper*
- Power modeling: *McPAT*

3. Approach

❖ Dynamic Scheduling Policy

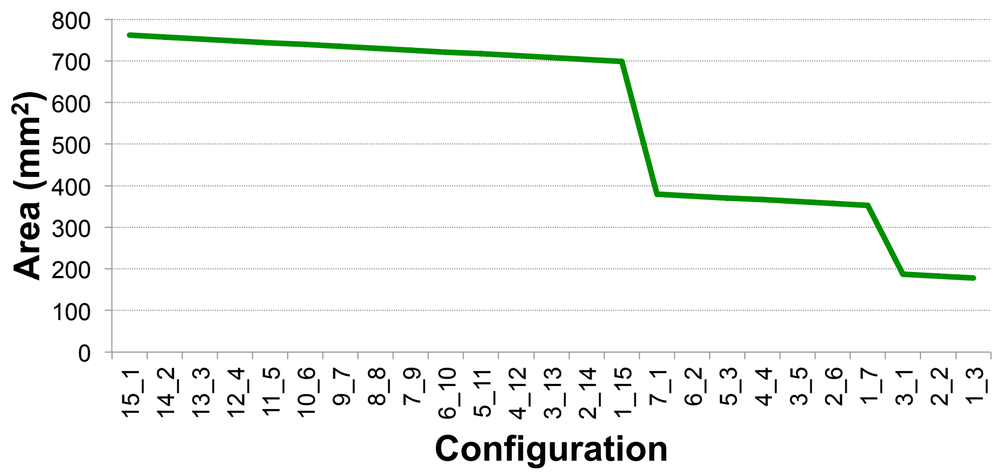
- Assumed the hardware consists of B big cores and S small cores.
- The adaptive scheduler selects a thread running on big core with lowest IPC (*thread_big*) and selects a thread running on small core with highest IPC (*thread_small*).
- If the IPC of *thread_big* is lesser than dynamic mean IPC of the big core, the thread is assigned to the small core and in case IPC of *thread_small* is greater than the current core's sum of mean and variance, it is moved to the big core.
- Each thread is pinned to a specific core and cores are handed out to new threads in round-robin fashion. If multiple threads share a core, they are time-shared with a configurable quantum.

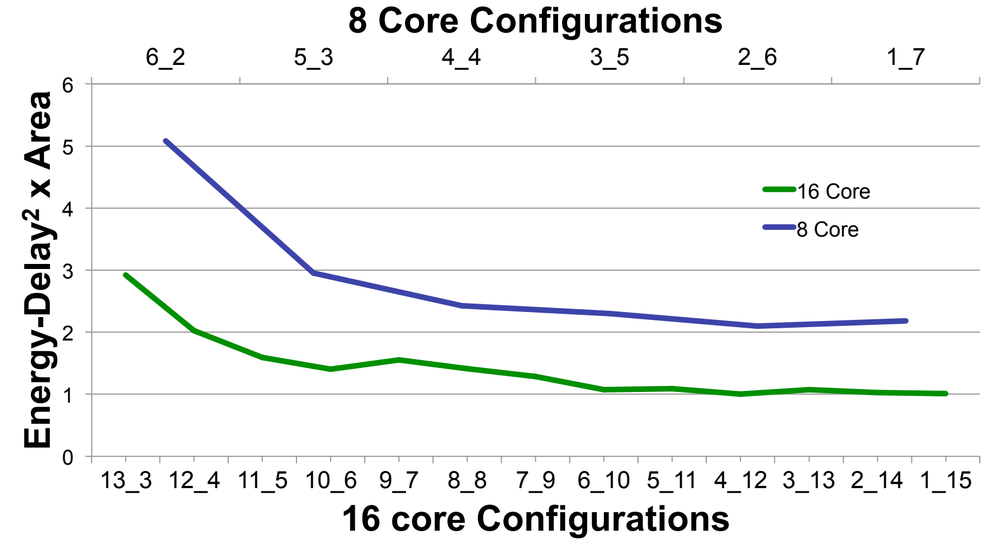
```
for core in NUM_CORES
    if (core is BIG) and (thread_core != INVALID)
        if (core_ipc_i < big_ipc)
            big_ipc = core_ipc
    if (core is SMALL) and (thread_core != INVALID)
        if (core_ipc_i > small_ipc)
            small_ipc = core_ipc

BIG_THRESHOLD = MEAN(big_core_ipc)
SMALL_THRESHOLD = MEAN(small_core_ipc) + VARIANCE(small_core_ipc)

if (big_ipc < BIG_THRESHOLD) and (small_ipc > SMALL_THRESHOLD)
    moveToSmall(thread_big)
    moveToBig(thread_small)
else if (big_ipc < BIG_THRESHOLD) and (small_ipc < SMALL_THRESHOLD)
    moveToSmall(thread_big)
```

4. Results





❖ Conclusion

- The new scheduler was able to optimize Energy-Delay^2 product based on dynamic IPC variations in the benchmark. Statistical analysis of IPC makes it adaptive to any application.
- Based on user's area budget, an appropriate configuration can be chosen using this model.
- Having ¼th of the total cores as 'big' cores proved to strike the "sweet-spot".