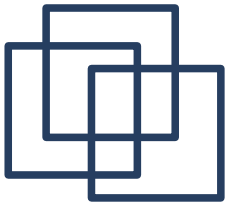
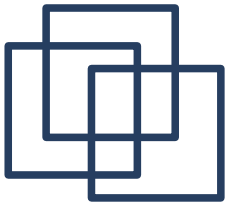


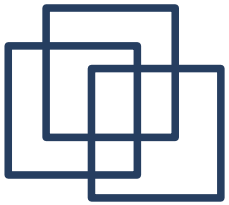
A Discussion of

High-Performance Throughput Computing



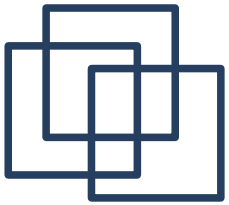
Throughput Computing

- In a computer system which can be running many tasks in parallel, higher performance can be obtained in the same area of silicon by using more smaller cores than fewer, complicated cores
- Usually, more simple cores will have lower power dissipation than the number of complicated cores that occupy the same amount of silicon



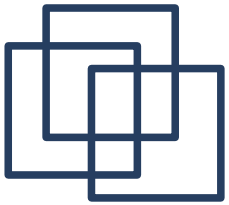
High-Performance

- There are still a number of single threaded applications which demand good single thread performance
- Ideally want high throughput (SMP and/or SMT) and high single thread performance



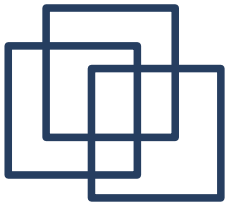
SMT

- Can keep processor busy while waiting for memory loads by sharing functional units among many threads
- Memory accesses can take the same amount of time as running through the rest of the pipeline many times
- Can overlap memory accesses and use the same functional units for a few threads



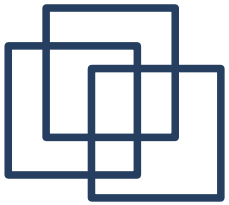
Hardware Scouting

- When a cache miss occurs, or any other event that would cause a stall, a checkpoint is placed at that point in the instruction stream
- Thread continues to execute, using a second set of registers so that it can continue executing without affecting program state
- Once memory load is complete, resume execution at checkpoint



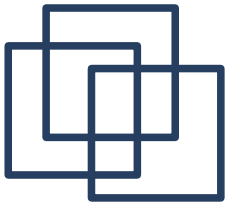
Advantages

- Data for future memory loads is placed into caches
- Ensures next instructions are loaded into cache
- Branch prediction improved
- Simple hardware scheme to hide memory access latencies
- Can use smaller caches



Disadvantages

- Only speeds up code which is affected by cache misses
- Difficult to implement with an out-of-order processor
- Won't improve situations where future memory loads or branches depend on a current load with a cache miss
- No future instructions are actually executed



Questions/Comments?