A self-improving work-stealing scheduler
Zafar Ahmad, Rezaul A. Chowdhury
IACS, Stony Brook University, NY 11794

Task Scheduling

**Goal:** Given a task graph and number of workers minimize the execution time.

**Challenges**
- An NP-Hard Problem
- Support full interoperability with serial binaries compiled to use a linear stack
- Enable a scheduler to achieve nearly perfect linear speedup on computations with ample parallelism
- Consume a bounded amount of stack space that is practical for general-purpose systems

Load Balancing

What happens when some workers exhaust their local task pools while others still have tasks left?

- Share some of the local tasks with underutilized workers
- Let idle workers take the initiative to “steal” some of the local tasks of other workers

The former is called **work sharing**, the latter **work stealing**.

Victim Selection

When any workers exhaust it’s local task pool it randomly select a victim and attempt to steal task from victim’s task pool. We use **Work-first scheduling** policy to achieve serial-parallel reciprocity, speed-up and stack-space bounds.

Potential Issues

- Random victim selection is not highly balanced.
- Scheduler doesn’t account the work size while scheduling.

Balanced Allocation

- When $n$ balls are thrown independently and uniformly at random into $n$ bins, the probability that the maximum load is more than $\frac{1}{n} \ln \ln n$ is at most $\frac{1}{n}$ for $n$ sufficiently large.
- With $d$ possible destination bins for each throw, if $d \geq 2$, maximum load is $\frac{\ln \ln n}{\ln d}$ with high probability.
- If $d > 2$ performance improves by only a constant factor; remains $\Theta(\ln \ln n)$.

Our victim selection policies

- Default – Select a random victim
- Based on Job count in victim’s task pool
  - Select a victim from two or more random choices
- Based on Workload in victim’s task pool
  - Select a victim from two or more random choices

Recursive Work Estimation

Optimization Challenges

- Introduces extra overheads –
  - Work estimation in base cases (timer polling overhead)
  - Work estimation in inner nodes (Regression analyses overhead)

Conclusion & Future Plan

- Improves cache misses, number of steals, page faults but running time remains the same due to extra overheads.
- Shows running time improvement for multi-run.
- Improving the work estimation scheme.
- Extending the design for distributed memory domain.