Impact of Objective Model on Search-based Parallel Computer Job Scheduling

Sangsuree Vasupongayya, Su-Hui Chiang, Bart Massey
{vsang,suhui,bart}@cs.pdx.edu

1. Parallel Job Scheduling

Properties of the scheduling system
- Similar to a rectangular Tetris without rotation
- Pick any piece from the stack of pieces (jobs)
- Online scheduling
  - Don’t know when the job will come
  - Don’t know the job size
- Non-preemptive
  - Cannot undo any previous decision
- All jobs must be scheduled

2. Motivation

Performance goals
- Typically there are multiple goals, e.g.,
  - prevent ‘starvation’
  - minimize average slowdown
  - maximize ‘fairness’
- But they potentially conflict with each other

Challenges
- How to optimize for multiple goals?
- How to define multiple goals in one objective?

3. Common approach

Based on predefined priority function
- Cannot specify objective
- Need tune priority function in ad-hoc manner
- Performance can be unexpected

Backfilling

4. Our Approach: Search-based Parallel Computer Job Scheduling

Objective model (Sec. 6,7)
- allows administrator to declaratively specify high-level performance goals
- must be intuitive and flexible

Scheduling engine (Sec. 5)
- employ combinatorial search to select jobs for execution

Modeling module (future work)
- collects workload and scheduler performance information
- wait-time, runtime, new arrivals prediction

5. Search Algorithms

- Organize all possible ordering in to a tree
- each path: an order of jobs for consideration
- n waiting jobs have n! ordering
- order of considering NOT the order the jobs can be started

Search
- Goal: find the path that optimize performance according to the objective
- Problem: time consuming
  - find good solutions within time constraint
- We found depth-bound discrepancy search (DDS) to perform well (see Cluster05)

6. Objective Model

Consider two goals commonly desired
1 prevent ‘starvation’
  → let measure excessive wait (ExW)
2 minimize average slowdown (X)

Intuitive models compared here
A) Hierarchical model, e.g.,
  - L1: minimize Tmn: total ExW
  - L2: minimize X

B) Explicit tradeoff model, e.g.,
  1) \( \Delta(T_{mn}) > 0 \) AND \( \Delta(X) \geq \gamma(X) \), OR
  2) \( \Delta(X) > 0 \) AND \( \Delta(T_{mn}) \geq \gamma(T_{mn}) \)

(\( \Delta \): improvement; \( \gamma \): degradation)

Other models studied, not shown.

7. Impact of Objective Model Results

Hierarchical vs Tradeoff Model

(a) Total ExW
(b) Avg. Slowdown
(c) Max. Wait
(d) Max. Slowdown

- LXF-backfill improves average slowdown but has worse maximum wait than that under FCFS-bf (Fig a,b)
- Search-based policies achieves best or close to the best for all measures except 1/04
- Hierarchical model has poor performance on slowdown measures (Fig. b,d) than that under Tradeoff model
  → Hierarchical improves high-level performance slightly at a huge expense on the low-level performance

- FCFS-bf: poor performance for wide jobs (N>32), even if they are short
- LXF-bf: improve short-wide jobs (T \leq 1h, N>32) but let long-large jobs (T>8h, N>8) suffer
- Hierarchical: improve short-wide jobs without sacrificing long-wide jobs as much as that under LXF-bf
- Tradeoff: improve short or small jobs further from that under Hierarchical

8. Conclusion

Search-based policies (using hierarchical or tradeoff objective model ) simultaneously beat traditional backfill policies (FCFS-bf and LXF-bf) w.r.t. the objective studied

Explicit tradeoff objective model shows potential to make a better tradeoff than hierarchical objective model