EXPLORATORY TESTING AT SCALE

Jiri Simsa, Randy Bryant, Garth Gibson (Carnegie Mellon University), Jason Hickey (Google)

PROBLEM

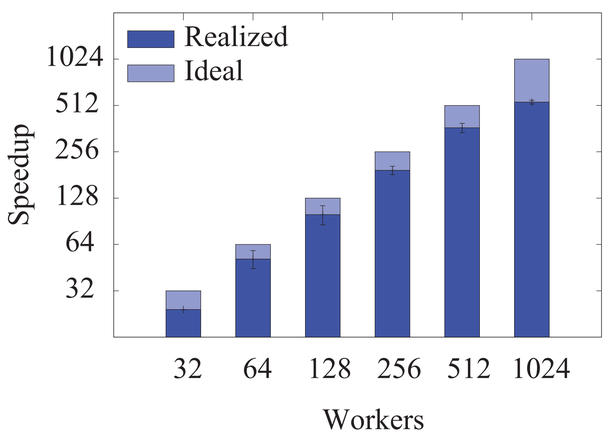
- Concurrency manifests test non-determinism
- Systematic testing controls the order of concurrent events and systematically enumerates different concurrent scenarios
- To mitigate combinatorial explosion, state space reduction techniques avoid exploration of equivalent scenarios
- For large-scale distributed programs, parallelize testing: efficient & load-balanced exploration

SOLUTION

- Distributed dynamic partial order reduction (DPOR) based on a novel state space exploration algorithm: n-partitioned depth-first search
- Dynamic deadline for fault tolerance; trade space complexity for parallelism in testing
- Implementation demonstrated to scale up to cluster of 1,024 nodes (speed up upwards of 750x)

SCALABILITY

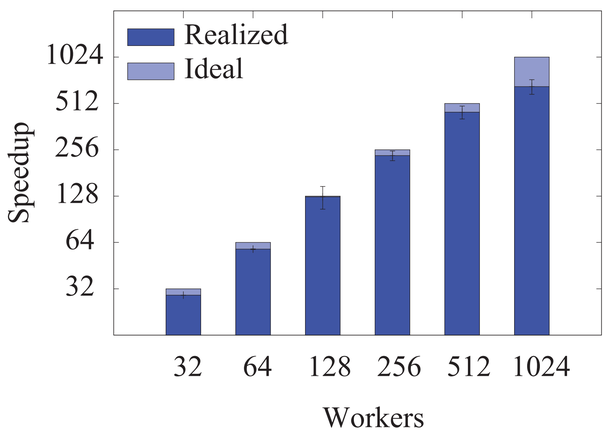
Resource(6,6)* with concurrent DPOR



Conf	Runtime(s)	Speedup
32 Workers	31,105.00	24.18
64 Workers	14,681.67	51.42
128 Workers	7,557.33	99.98
256 Workers	3,874.67	193.88
512 Workers	2,051.67	366.30
1024 Workers	1,396.33	538.49

For this example, dynamic partial order reduction explores on the order of 18.5 million branches and the sequential implementation is expected to require 209 hours to finish.

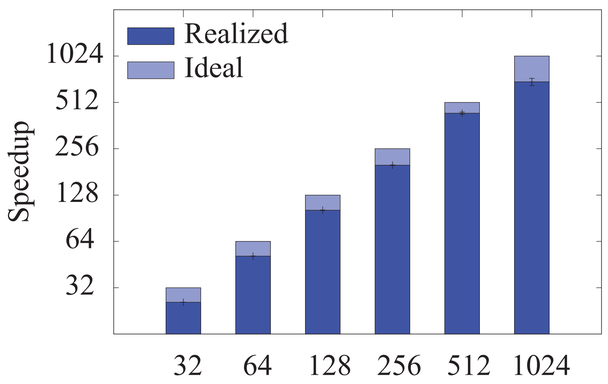
Store(12,3,3)* with concurrent DPOR



Conf	Runtime(s)	Speedup
32 Workers	26,646.00	29.06
64 Workers	13,396.33	57.80
128 Workers	6,252.33	126.09
256 Workers	3,332.00	233.28
512 Workers	1,784.00	434.54
1024 Workers	1,269.33	610.61

For this example, dynamic partial order reduction explores on the order of 21 million branches and the sequential implementation is expected to require 215 hours to finish.

Scheduling(10)* with concurrent DPOR

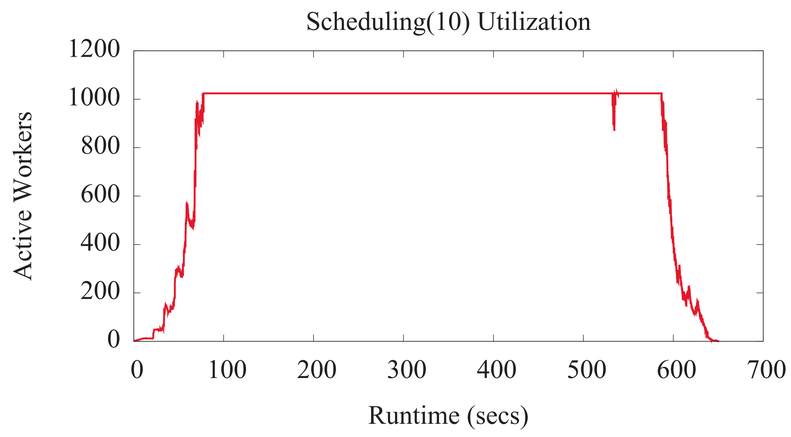


Conf	Runtime(s)	Speedup
32 Workers	17,707.33	25.73
64 Workers	8,870.67	51.36
128 Workers	4,468.33	101.96
256 Workers	2,278.33	199.98
512 Workers	1,046.00	435.67
1024 Workers	655.67	696.21

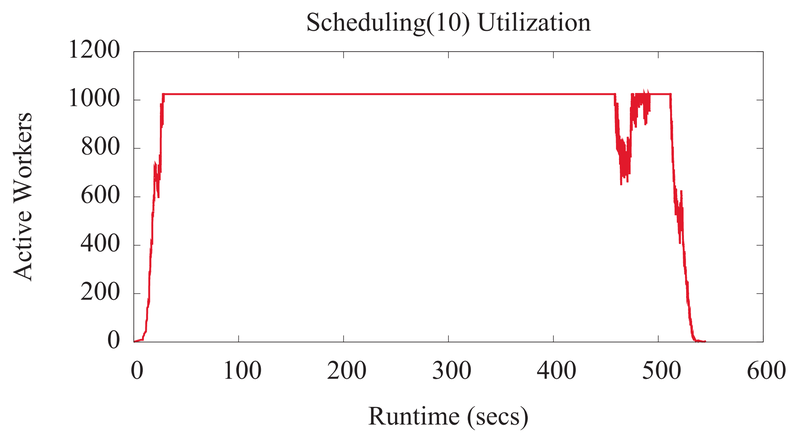
For this example, dynamic partial order reduction explores on the order of 3.6 million branches and the sequential implementation is expected to require 126 hours to finish.

IMPROVING UTILIZATION

- Initial and final phase of exploration underutilizes worker fleet at large scale
- Two techniques to improve utilization: variable time budget, redundant exploration



1024 workers, without optimizations.



1024 workers, with optimizations.

- Improved speed up from 538x to 916x, from 610x to 759x, and from 696x to 865x for the respective tests

THEORETICAL LIMITS

- The centralized master is a potential bottleneck
- At 1024 workers, memory overhead of the master 4MB → possible to scale to hundreds of thousands of workers
- At 1024 workers and 10-second worker time budget, master processes roughly 100 RPCs per second at 20% CPU utilization
- To scale beyond thousands of workers → scale worker time budget, better hardware, optimize software stack

*instances of actor program tests from the test suite of Google's next-generation cluster management system

Intel Science & Technology Center for Cloud Computing
Google
Center for Computational Thinking Carnegie Mellon
Sponsored by Microsoft Research
Carnegie Mellon University
Georgia Tech
intel
Princeton University
UC Berkeley