What Test Should I Run?

Improving Concurrency Tests using State Space Estimation

Ben Blum

Jiří Šimša, Garth Gibson

Parallel Data Laboratory
Carnegie Mellon University

Outline

Systematic Testing

- Introduction and motivation
- State space estimation

Using State Space Estimates

- Improving estimation quality
- Automatic input refinement for small tests

Conclusion

Example

```
int thread_fork()
{
    thread_t *child = spawn_new_thread();
    add_to_runqueue(child);
    return child->tid;
}
```

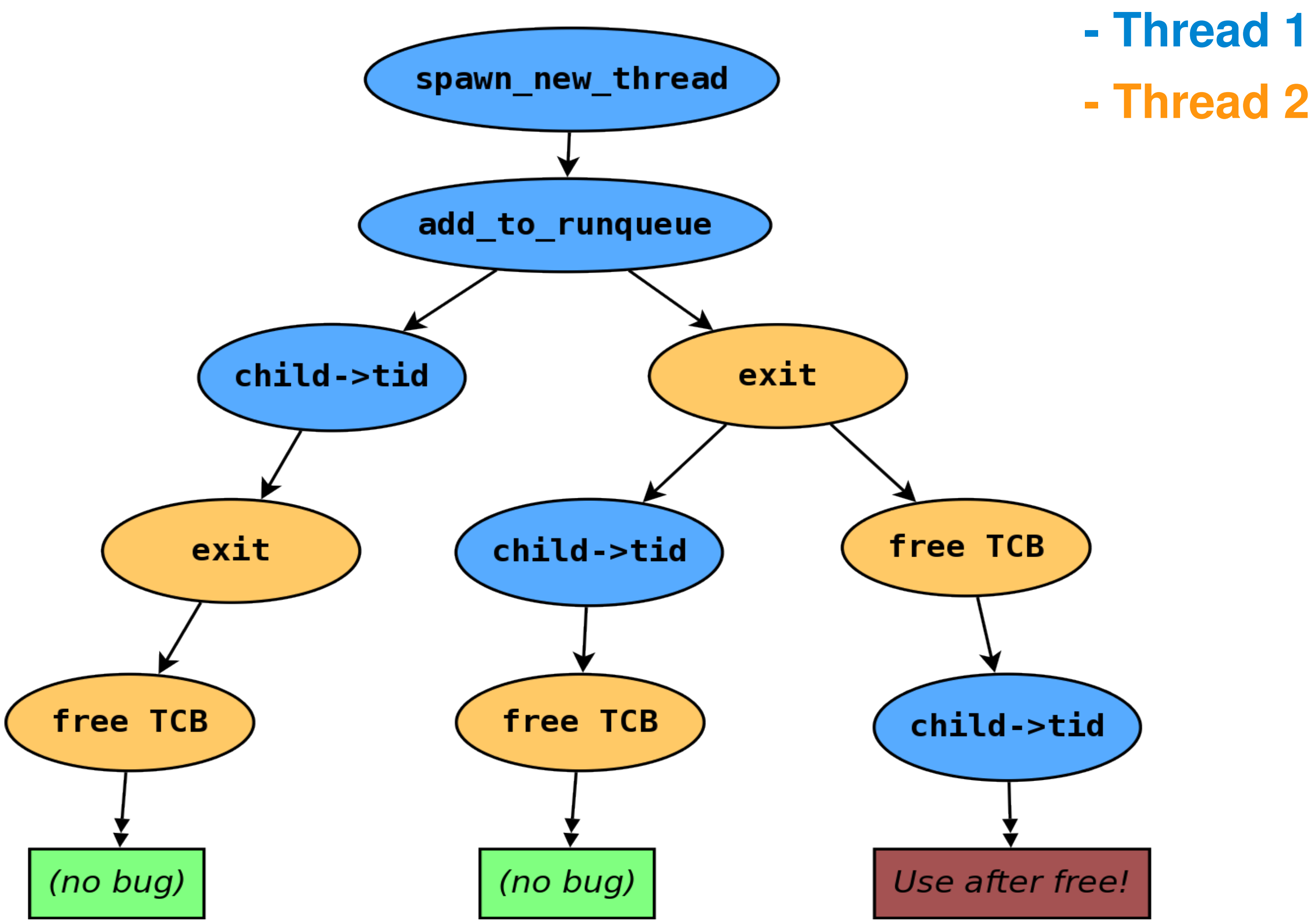
Example

```
int thread_fork()
{
    thread_t *child = spawn_new_thread();
    add_to_runqueue(child);
    return child->tid;
}
```

"child" gets freed!

- On exit, child's state is freed
- Forking thread does use-after-free
- Might return garbage instead of thread ID

The Execution Tree



Systematic Testing

Systematic (exploratory) testing *[Godefroid '97]*

- Exhaustive search of a concurrent test's state space
- State space defined by “decision points”
- dBug *[Simsa '11]*, Landslide (for kernels) *[Blum '12]*

Systematic Testing

Systematic (exploratory) testing *[Godefroid '97]*

- Exhaustive search of a concurrent test's state space
- State space defined by “decision points”
- dBug *[Simsa '11]*, Landslide (for kernels) *[Blum '12]*

Problem: Many state spaces are too large to test feasibly.

- Exponential trade-off: thoroughness versus time
- Reduction techniques help (somewhat) *[Flanagan '05]*

Can we approach this trade-off automatically?

State Space Estimation

On-line estimation can guess total exploration time.

- Enables resource allocation for scheduling many tests to maximize completion *[Simsa '12]*
- As exploration progresses, estimate updates to reflect new knowledge about the state space.
  - Assumes similar structure across state space.
  - However, pruning interferes with estimation.

State Space Estimation

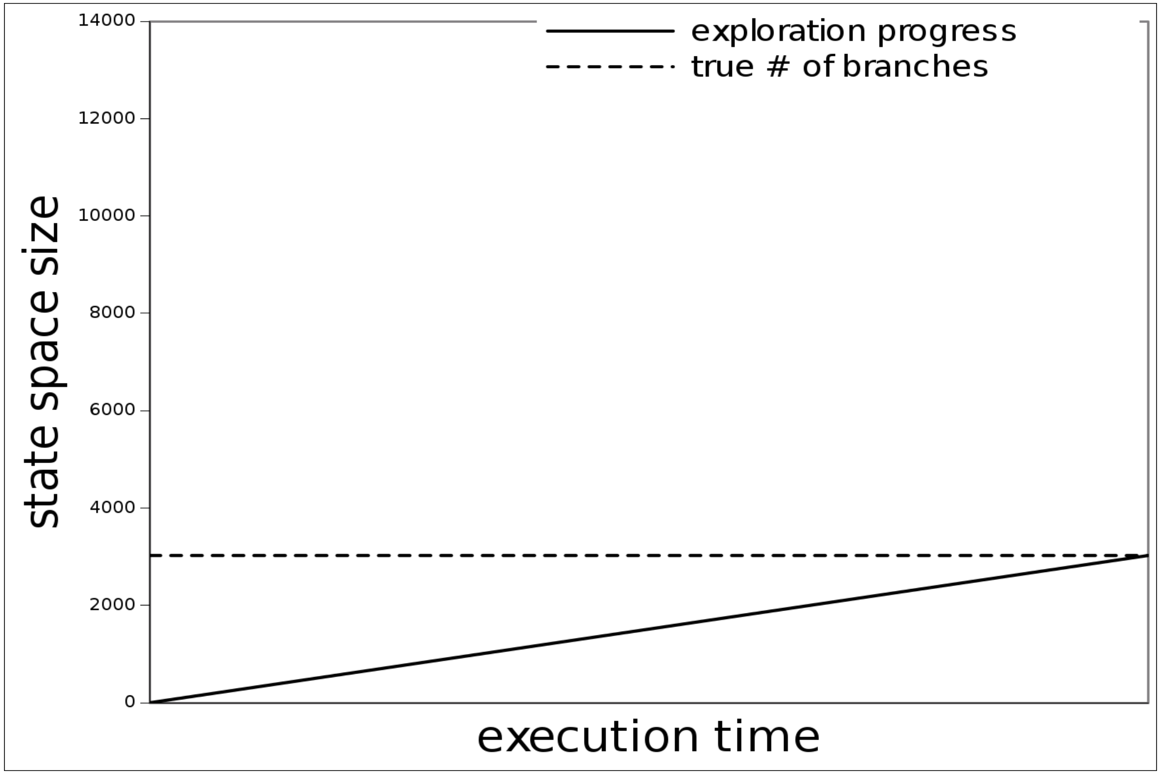
Why is state space estimation difficult?

- Estimation varies as time progresses
- Estimation varies across exploration orderings

State Space Estimation

Why is state space estimation difficult?

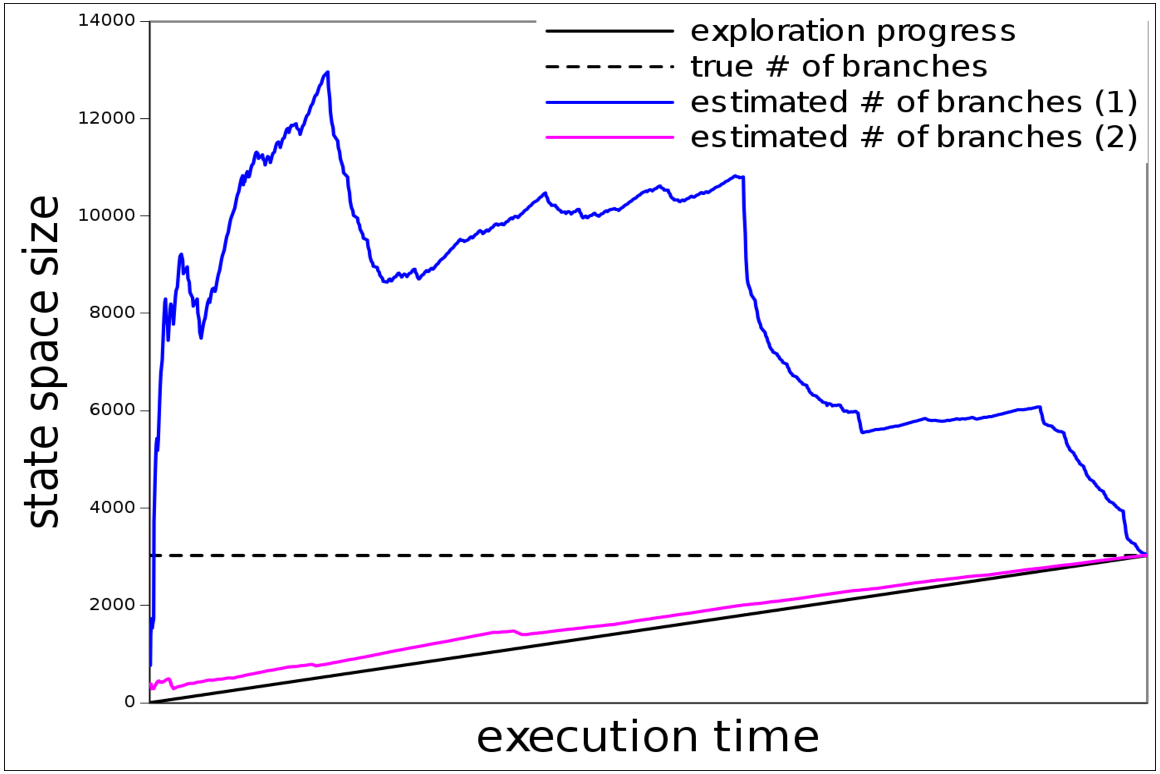
- Estimation varies as time progresses
- Estimation varies across exploration orderings



State Space Estimation

Why is state space estimation difficult?

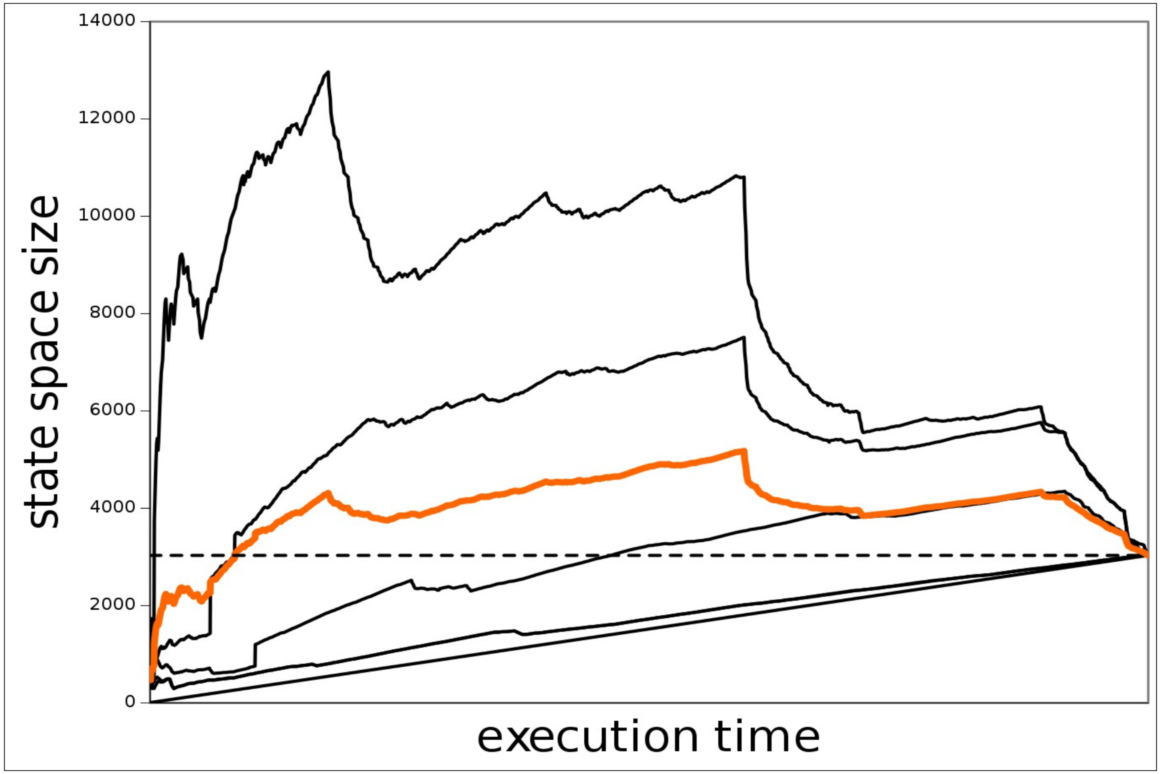
- Estimation varies as time progresses
- Estimation varies across exploration orderings



Improving Estimation Quality

Sample estimates from many parts of the tree at once.

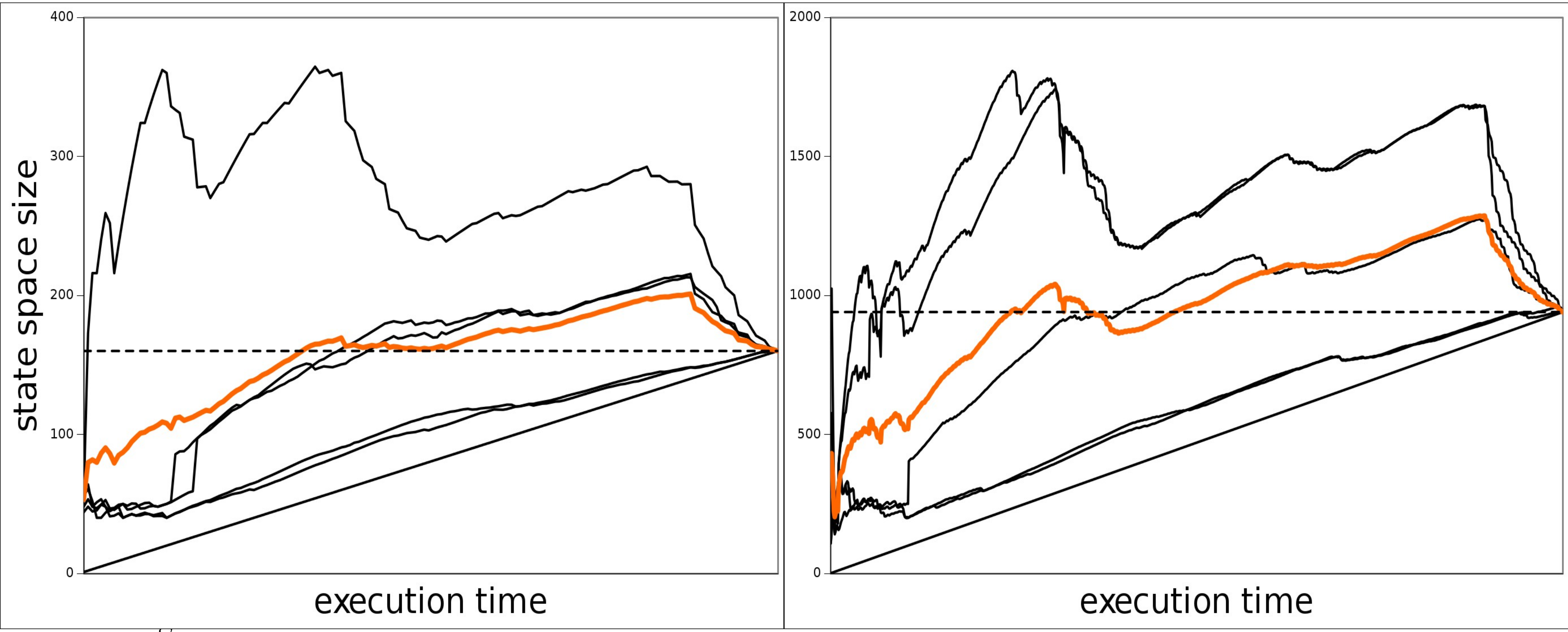
- Try different, random exploration orderings in parallel.
- Average of all estimates tends to be most accurate.



Improving Estimation Quality

Sample estimates from many parts of the tree at once.

- Try different, random exploration orderings in parallel.
- Average of all estimates tends to be most accurate.



Improving Estimation Quality

How to get the best estimate with a given CPU budget?

- Use value of estimate after exploring N branches.
- Exploring M ways in parallel requires lowering N.

What is a good heuristic for N versus M?

Test Case Refinement

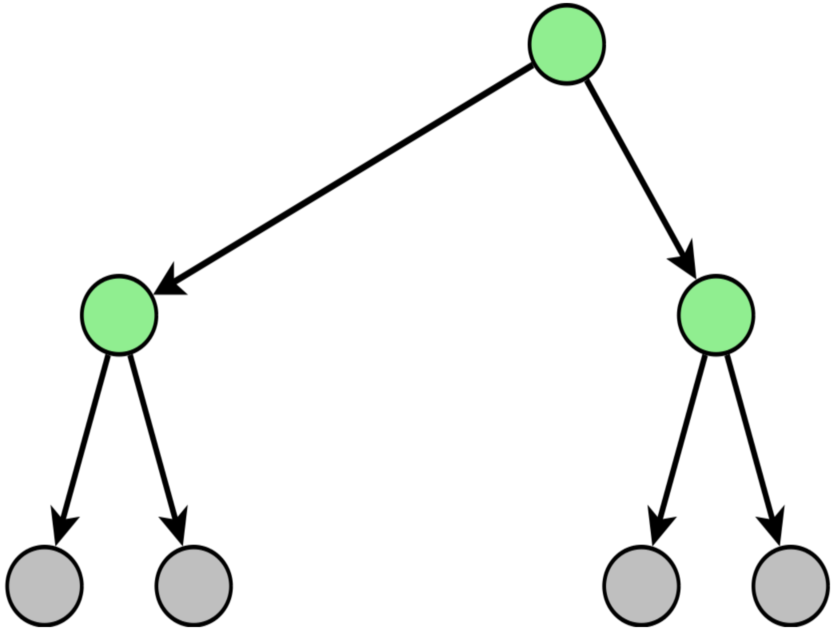
How can we create a test with a reasonable state space?

- User studies with 15-410 (operating systems)
  - Students worked best with an iterative process
  - "Start small, then add more decision points"

Test Case Refinement

How can we create a test with a reasonable state space?

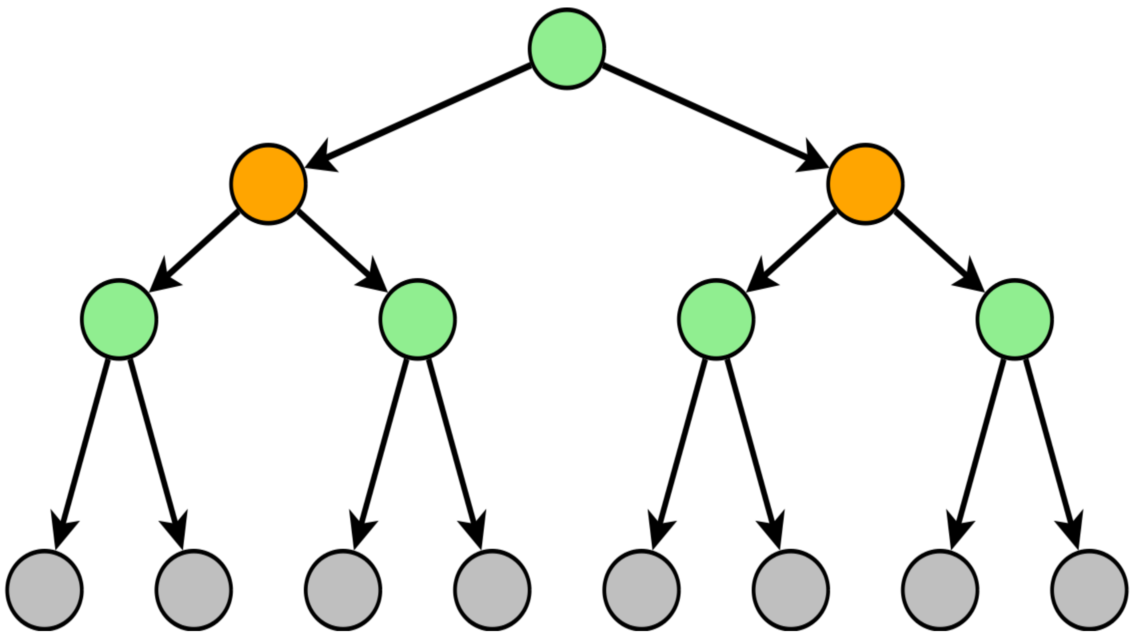
- User studies with 15-410 (operating systems)
  - Students worked best with an iterative process
  - "Start small, then add more decision points"



Test Case Refinement

How can we create a test with a reasonable state space?

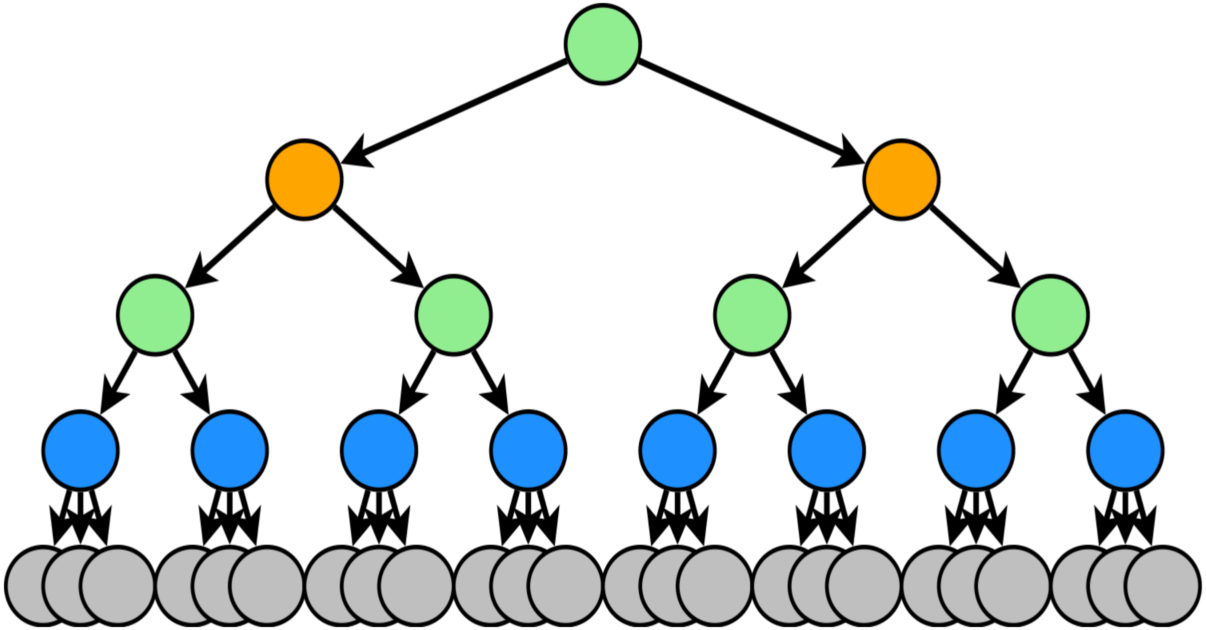
- User studies with 15-410 (operating systems)
  - Students worked best with an iterative process
  - "Start small, then add more decision points"



Test Case Refinement

How can we create a test with a reasonable state space?

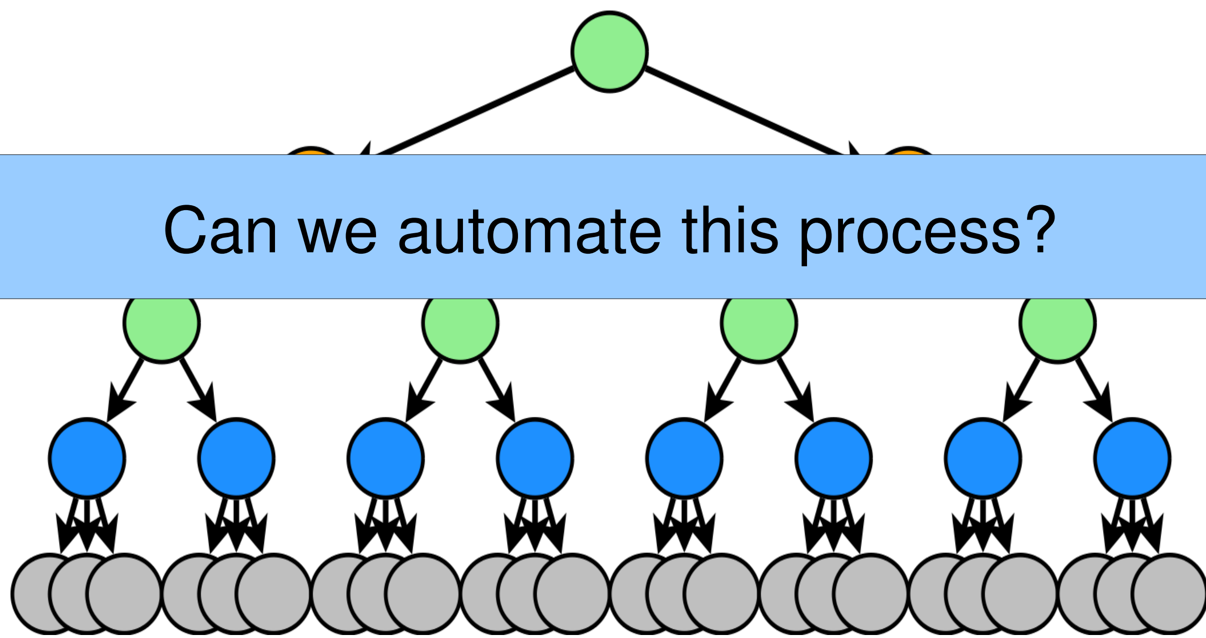
- User studies with 15-410 (operating systems)
  - Students worked best with an iterative process
  - "Start small, then add more decision points"



Test Case Refinement

How can we create a test with a reasonable state space?

- User studies with 15-410 (operating systems)
  - Students worked best with an iterative process
  - “Start small, then add more decision points”



Can we automate this process?

“Iterative Deepening”

Goal: Heuristically add as many decision points as possible for a given CPU time budget.

- Challenge: decision points should be meaningful.
  - Lock/unlock calls (worked well with 15-410)
  - Analyze memory accesses to find data races
- Estimates let us judge state spaces as “too big”.
  - Test framework can test increasingly large sets of decision points until time runs out.

Conclusion

On-line estimation for concurrency tests is hard.

- Estimates vary dramatically with test configuration
- Can improve estimation quality with heuristic averages

Accurate estimates are important.

- With a large test suite, allows us to decide which tests are best to run
- With a single test, allows us to iteratively refine the test configuration until resources are exhausted

References

[Godefroid '97]

- Patrice Godefroid. VeriSoft: A Tool for the Automatic Analysis of Concurrent Reactive Software. CAV 1997.

[Flanagan '05]

- Cormac Flanagan and Patrice Godefroid. Dynamic partial-order reduction for model checking software. POPL 2005.

[Simsa '11]

- Jirí Simsa, Randy Bryant, Garth A. Gibson: dBug: Systematic Testing of Unmodified Distributed and Multi-threaded Systems. SPIN 2011.

[Blum '12]

- Ben Blum. Landslide: Systematic Dynamic Race Detection in Kernel Space. CMU-CS-12-118. May 2012.

[Simsa '12]

- Jiri Simsa. Runtime Estimation and Resource Allocation for Concurrency Testing. CMU-PDL-12-113. December 2012.

Related Work

Systematic testing

- MaceMC (NSDI '07) – liveness, random walking
- CHESS (PLDI '07) – iterative context bounding
- MoDist (NSDI '09) – network/disk model checking
- dBug (SSV '10) – dynamic partial order reduction
- SimTester (VEE '12) – interrupt injection, drivers

Data race detection

- Eraser (TOCS '97) – lock-set tracking, annotations
- DataCollider (OSDI '10) – random sampling, kernel
- RacePro (SOSP '11) – inter-process races