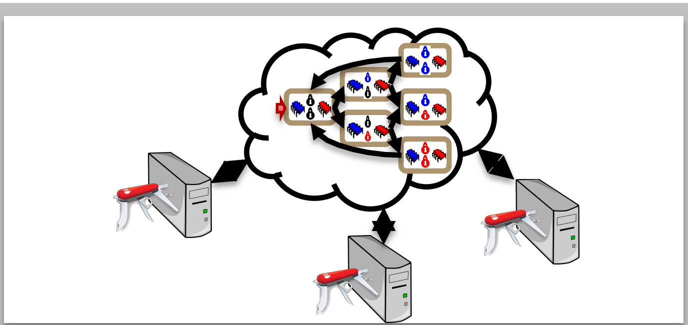
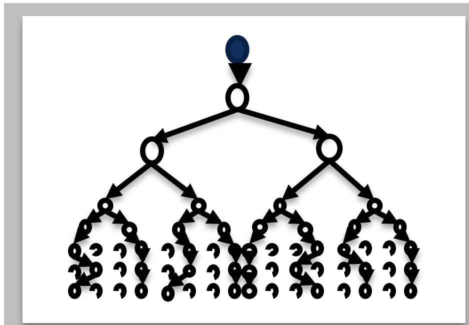
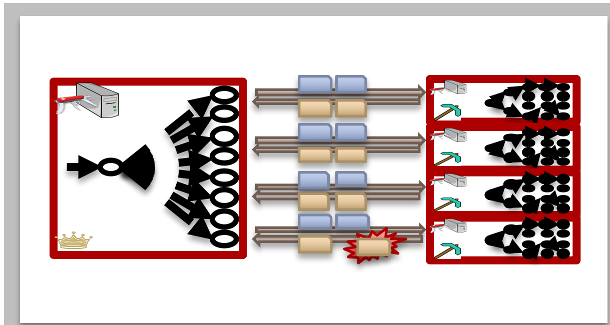
Exceptional service in the national interest

Sandia National Laboratories







Scalable Parallel Model Checking via Monte-Carlo Tree Search

Reed Milewicz and Simon Poulding

October 2017

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Mini-Introduction

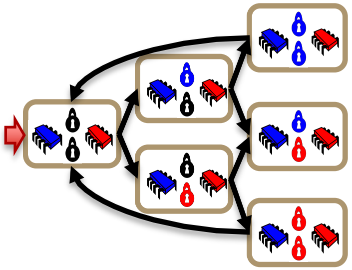
Sandia National Laboratories

- Who am I?
- Teaser

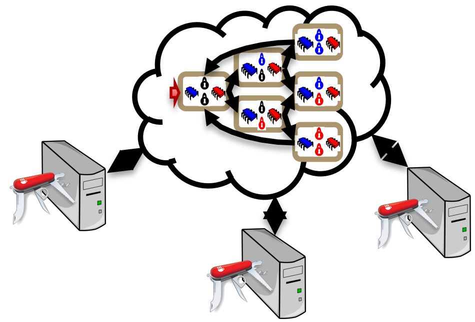
Who am I?

- Postdoctoral researcher at Sandia National Laboratories
- My areas of interest
  - Software Engineering
  - Formal Verification
  - Static and Dynamic Analysis
  - Source-to-Source Transformation Systems

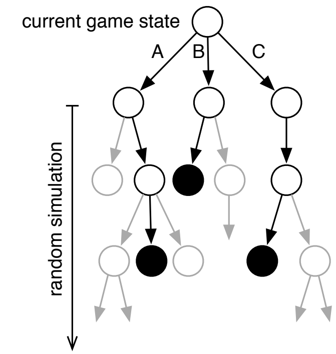
The Presentation on One Slide



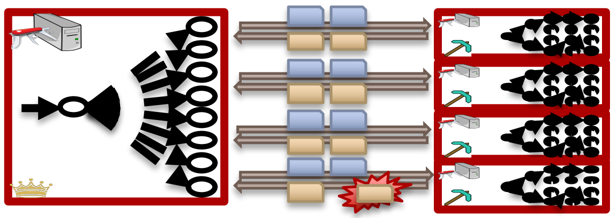
(1) Directed model checking is a powerful strategy for finding bugs in applications of interest.



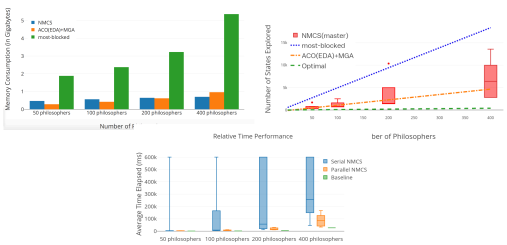
(2) Parallelizing any graph algorithm is hard, especially for things like DMC.



(3) Monte Carlo Tree Search is an efficient technique for navigating large state spaces.



(3) We present an adaptation of parallel MCTS for DMC.



(4) We share experimental results, and outline a path for future work.

Overview

- Background; Research Questions
- Design; Optimizations
- Implementation
- Experimental Results
- Future Work

Background

Background: Parallel Search

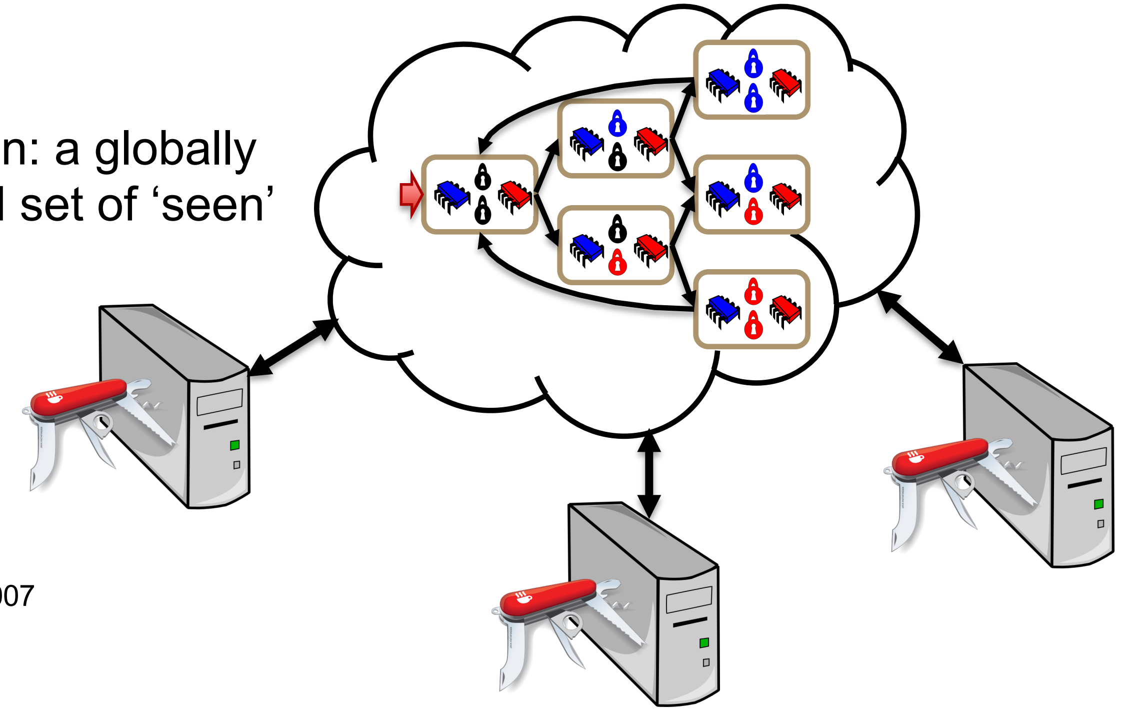
- In general, writing good parallel graph search algorithms is hard. Ideally, you want...
  - Data reuse for spatial/temporal locality
  - Cache-friendly
  - Limited communication
- As we move towards exascale...
  - Data movement will be **more expensive** relative to processing.
  - Memory per processor is actually expected to **decrease**.

> In the coming years we can expect systems with very large memory sizes, and increasing numbers of CPU cores, but with each core running at a relatively low speed.
>
> – Holzmann, Joshi, Groce 2008 (Swarm Verification)

Background: Parallel Search

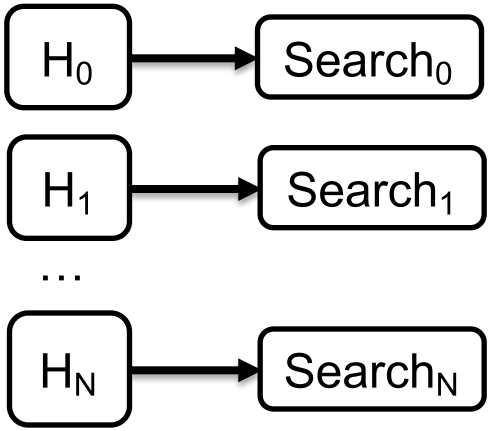
- Research on parallel model checking in the 2000s found that communication overhead would be a persistent problem[1,2].
- Ideally, we would like to limit redundant computation of states so as to maximize the amount of useful parallel work.

Naïve solution: a globally synchronized set of 'seen' states.

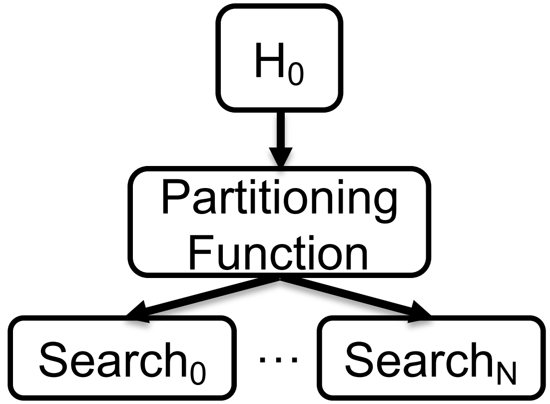


[1] Holzmann and Bosnacki 2007
[2] Stern and Dill 2001

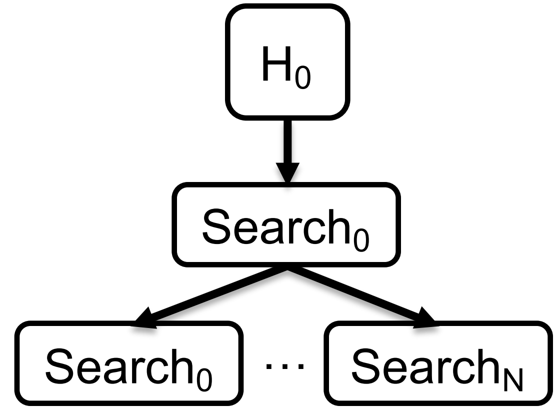
Background: Parallel Search



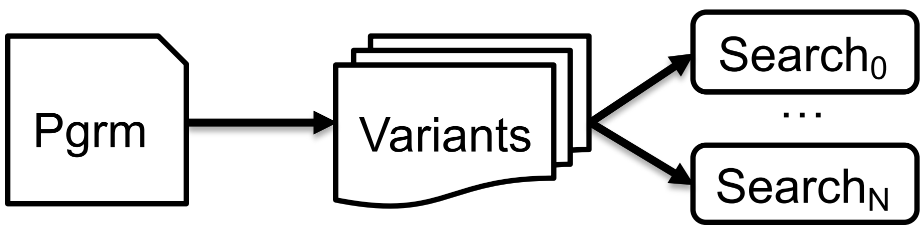
Many independent searches
(Dwyer et al. 2007,
Holzmann, Joshi, and Groce 2008)



Static partitioning
(Staats and Păsăreanu 2010)



Dynamic, on-the-fly partitioning
(Funes, Siddiqui, and Kurshid 2012)



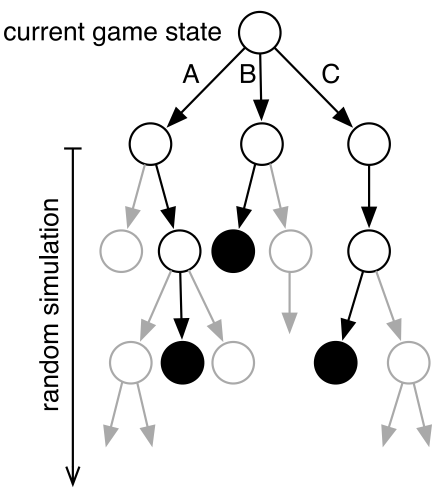
Task competition
(Nguyen et al. at ASE 2017!)

- Many works have followed that have introduced non-communicating or minimally communicating parallel search in this domain.

Background: MCTS

Sandia National Laboratories

- Monte-Carlo Tree Search (MCTS) algorithms are very popular within the game-playing AI community, most recently enabling the triumph of AlphaGo over world-class Go player Lee Sedol in 2016, a major milestone in the history of AI.
  - Anytime algorithm
  - Effectiveness of Monte Carlo random sampling is well-established.
  - Can be combined with [(meta-)*|(hyper-)*]heuristic search.
  - Memory efficient and readily parallelizable.

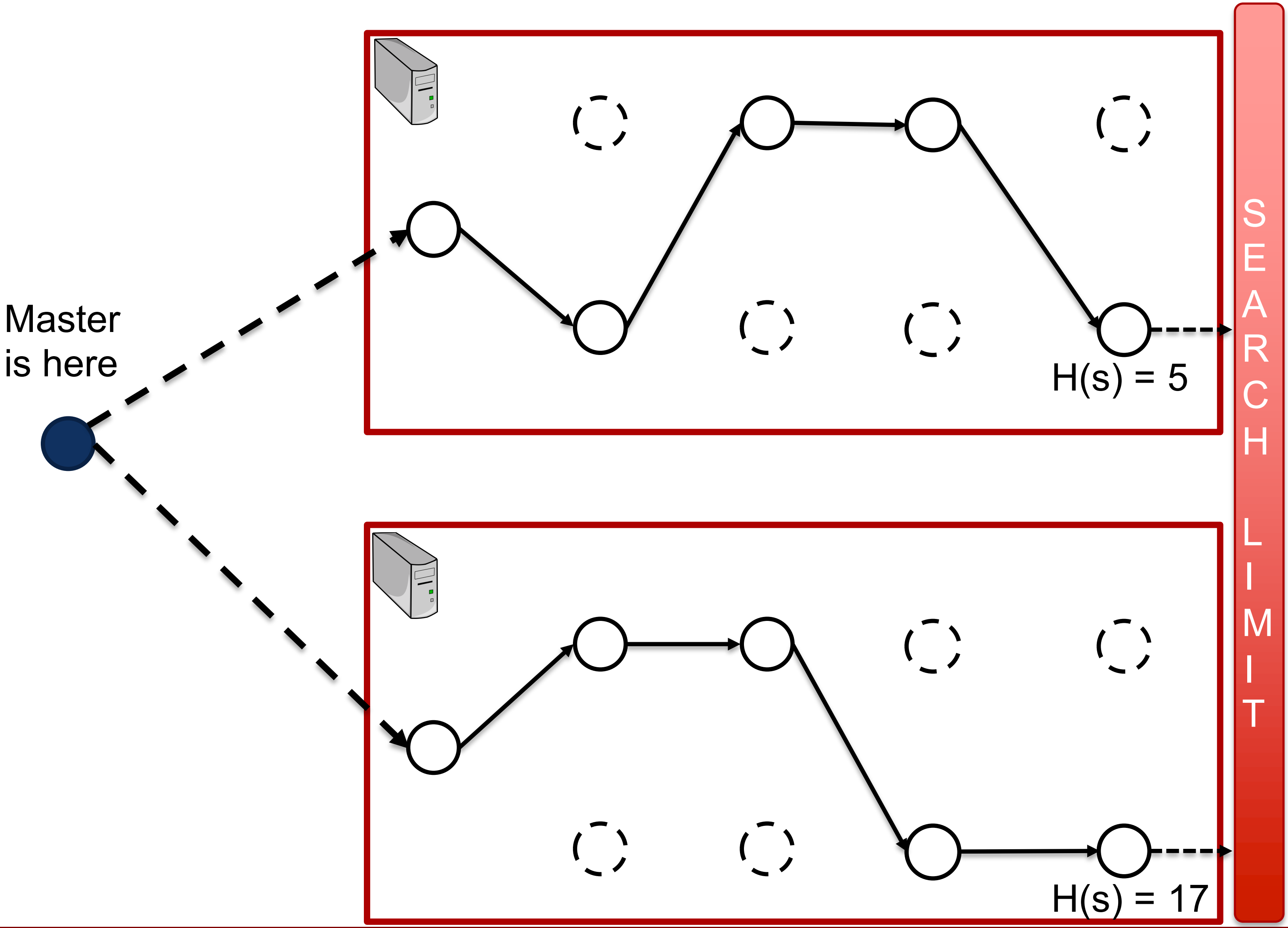


Prior Work

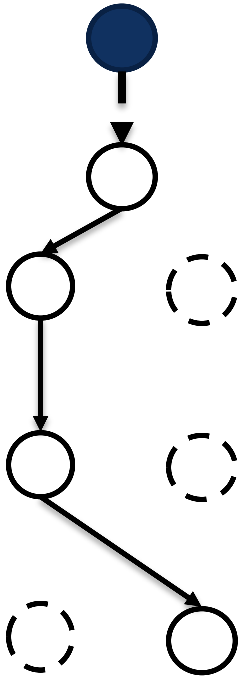
- Poulding and Feldt 2015 applied a variant of MCTS, Nested Monte Carlo Search, to the problem of directed model checking.
  - The implementation wasn't very efficient.
  - The paper itself was written in 48 hours.
  - The authors only considered the serial case.
  - But the results were very promising!
- In this work, we present…
  - A minimally communicating parallel MCTS search strategy.
  - Experimental results which demonstrate the effectiveness of our approach.

Design and Implementation

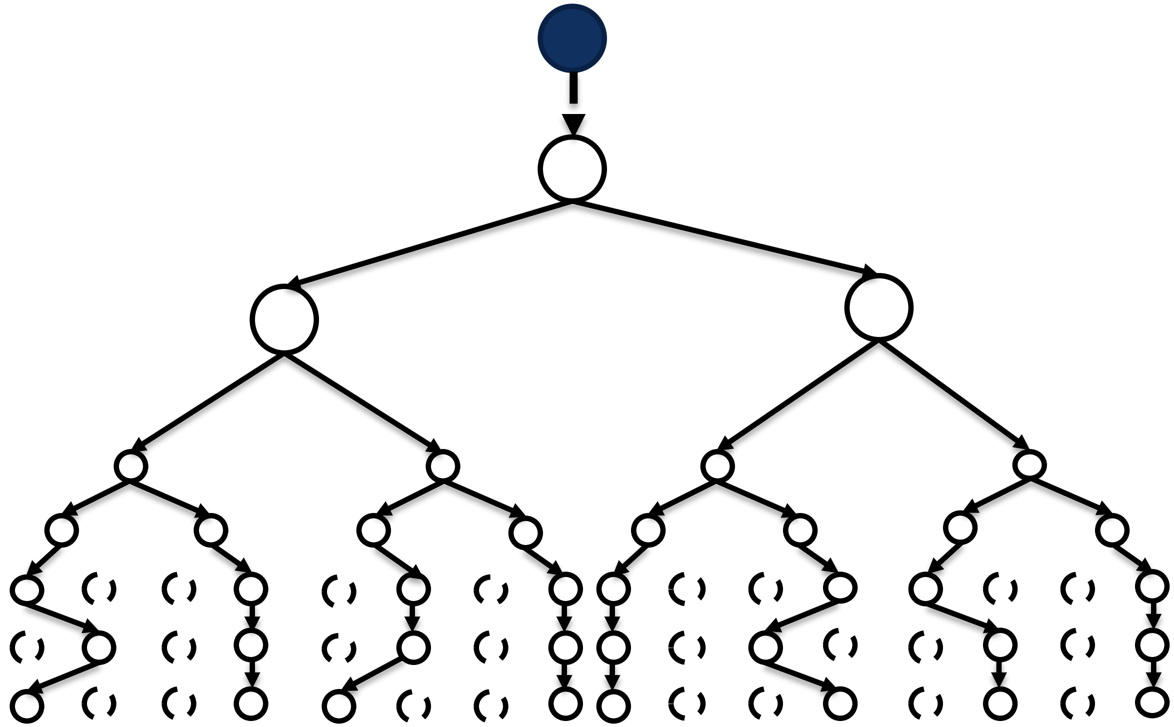
Local Search ala MCTS



Nesting MCTS



Nesting Level 0



Nesting Level 2

- MCTS can be recursively nested.
- Note: The workers can cache the states that they have explored, but it is not strictly necessary. The master is not responsible for retaining these states.

NMCS Optimizations

$$O(h^{N+1}b^{N})$$

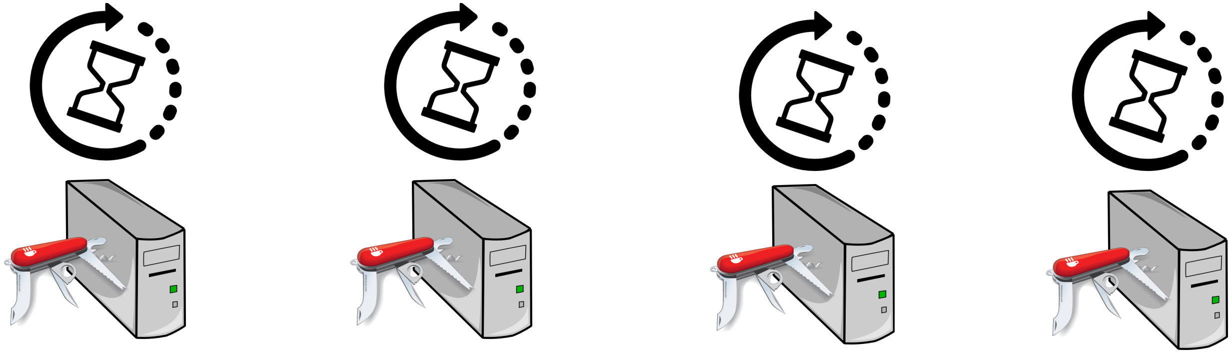
Where

N is the nesting factor
h is the depth of the tree
b is the branching factor

- In the domain of conventional games…
  - Computation of an individual state tends to be cheap.
  - Depth and branching factor are well-known in advance.

NMCS Optimizations

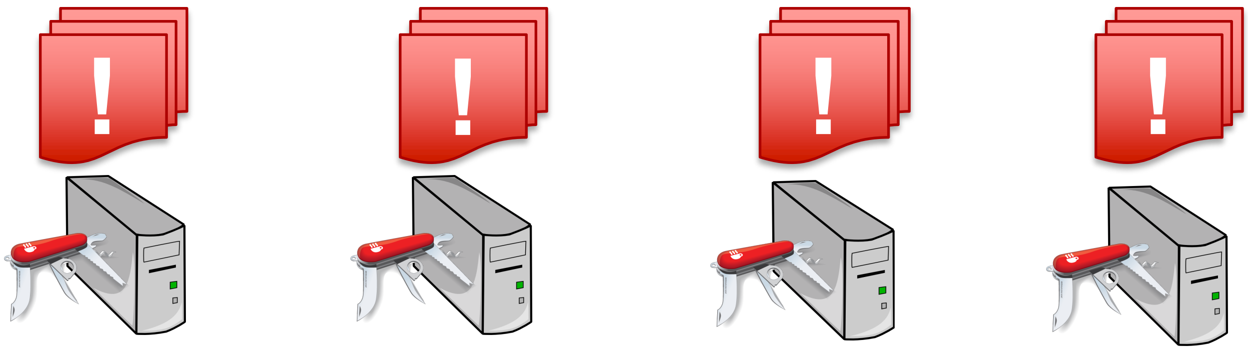
$$O(h^{N+1}b^{N}) \rightarrow O(l(h)^{N+1}b^{N})$$



- The search space can be **very** deep, and the computational cost of a state can be **very** high, which means that we can end up waiting for a long time on a worker to complete a simulation. We add a limit parameter here.

NMCS Optimizations

$$O(l(h)^{N+1}b^N) \rightarrow O(l(h)^{N+1}l(b)^N)$$



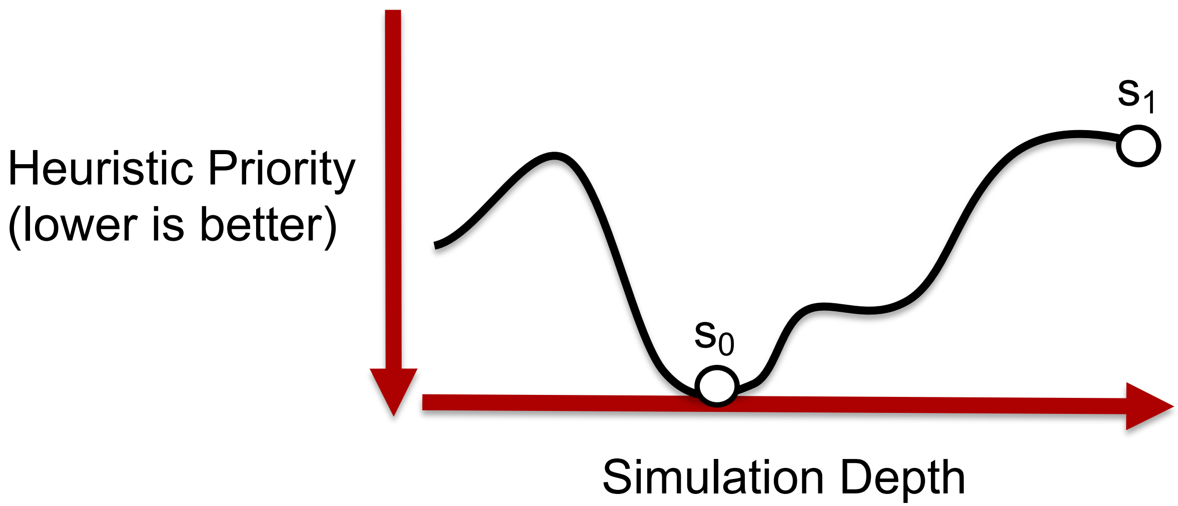
- The branching factor can grow to be extremely large. This means that each worker gets saddled with a backlog of simulations to perform. We also add a limit parameter here.

NMCS Optimizations

- In conventional NMCS, a simulation reports the heuristic value of the last state visited.
- In our implementation, we report the best value across all states visited.

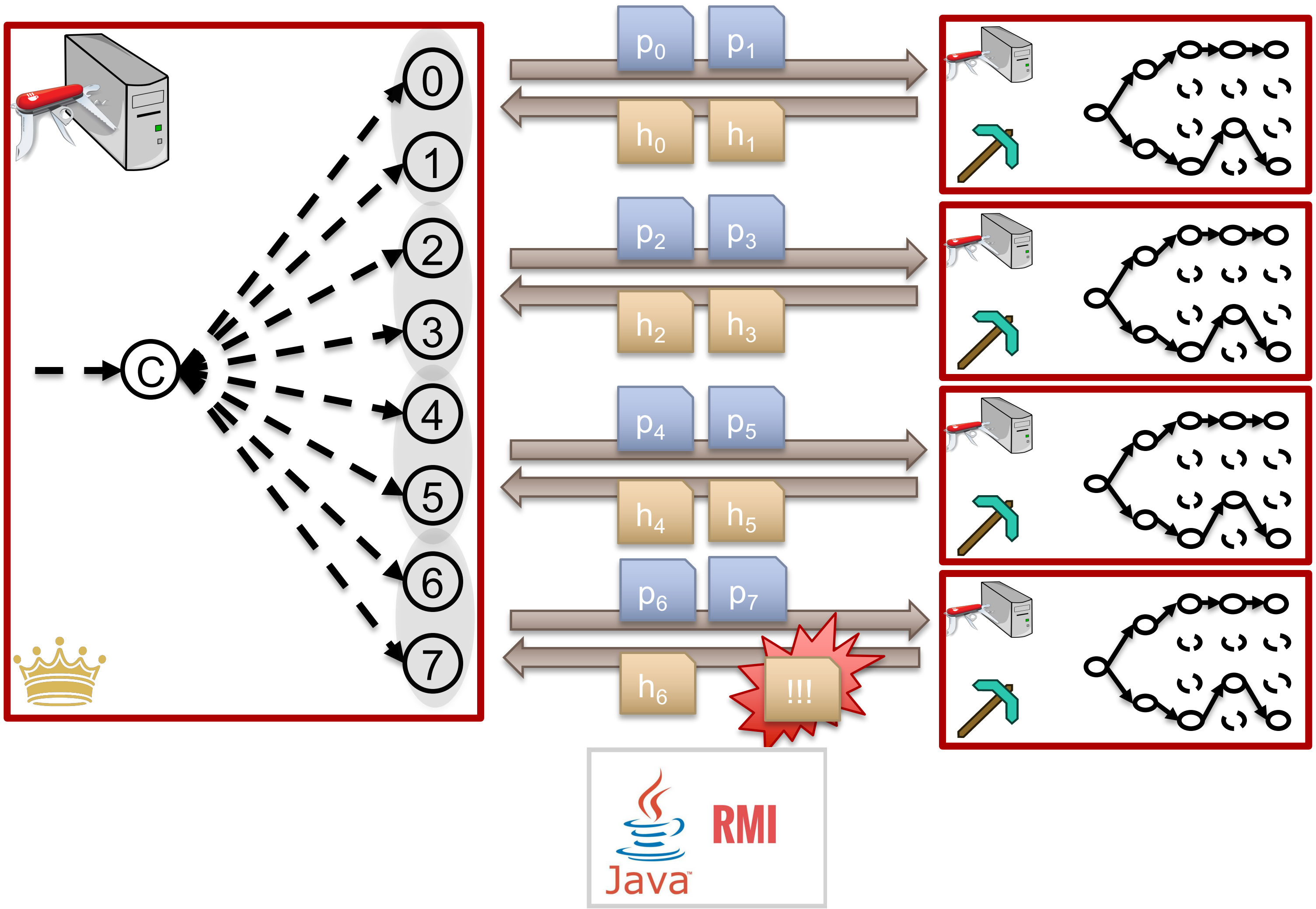
Anonymous Review #1

[…] That is a GREAT idea and on that alone I would accept this paper.



The state s_0 had the best heuristic value. The state does not contain a violation, but could be in the neighborhood of a violation.

Implementation



Evaluation

Evaluation

- **Benchmark**: Dining Philosophers Problem {50, 100, 200, 400 threads}, average of 10 runs, time limit of 600 seconds
- **Platform**: Eight 20-core Intel Xeon E5-2698 v4 nodes each with 32 GB of available RAM

Serial Baseline

$h_{mostblocked} = N_{alive} - N_{runnable}$ [1]

"Kitchen Sink" Serial Optimization

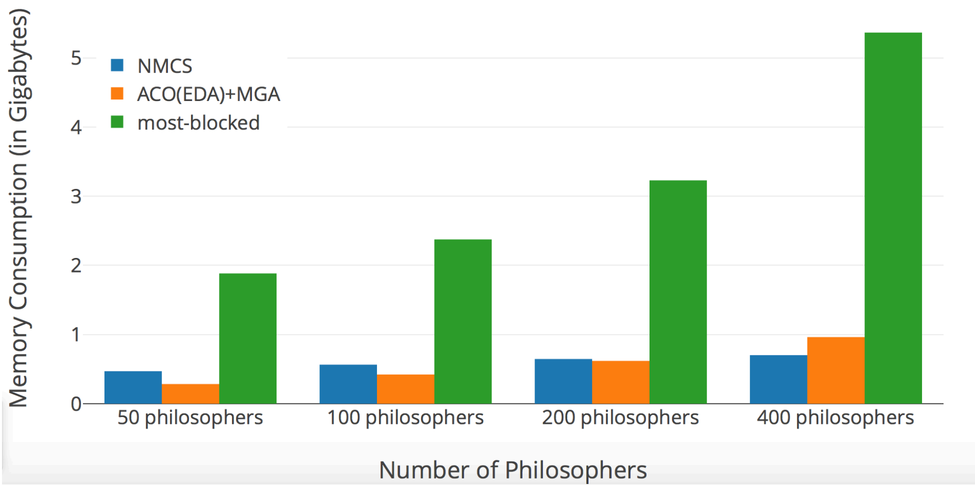
- EDA($h_{mostblocked}$)[2]
- ACO(EDA)[3]
- MGA(ACO)

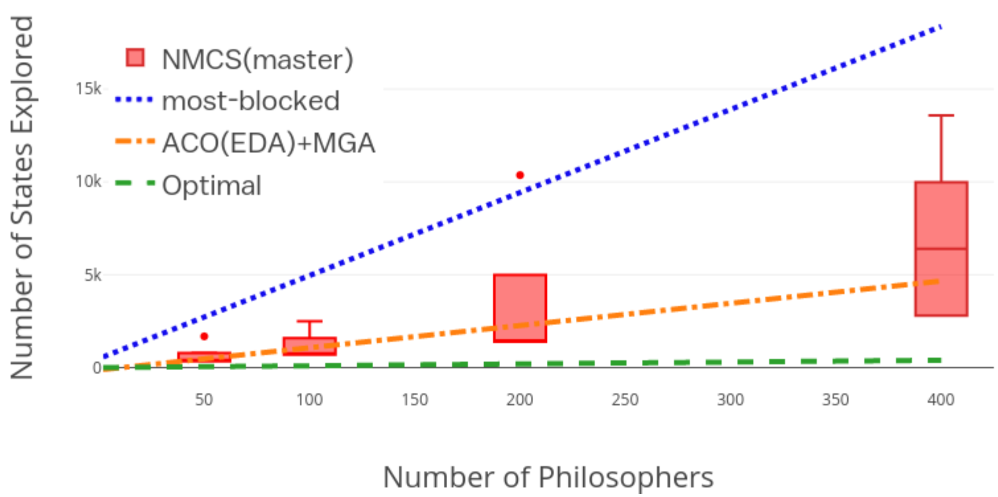
Parallel NMCS

- NMCS($h_{mostblocked}$)
- 1 master, 4 workers
- Nesting limit: {1}
- Depth limit: {4,8,16}
- Choice limit: {4,8,16}

[1] Groce and Visser 2002
[2] Staunton and Clark 2010
[3] Chicano and Alba 2008

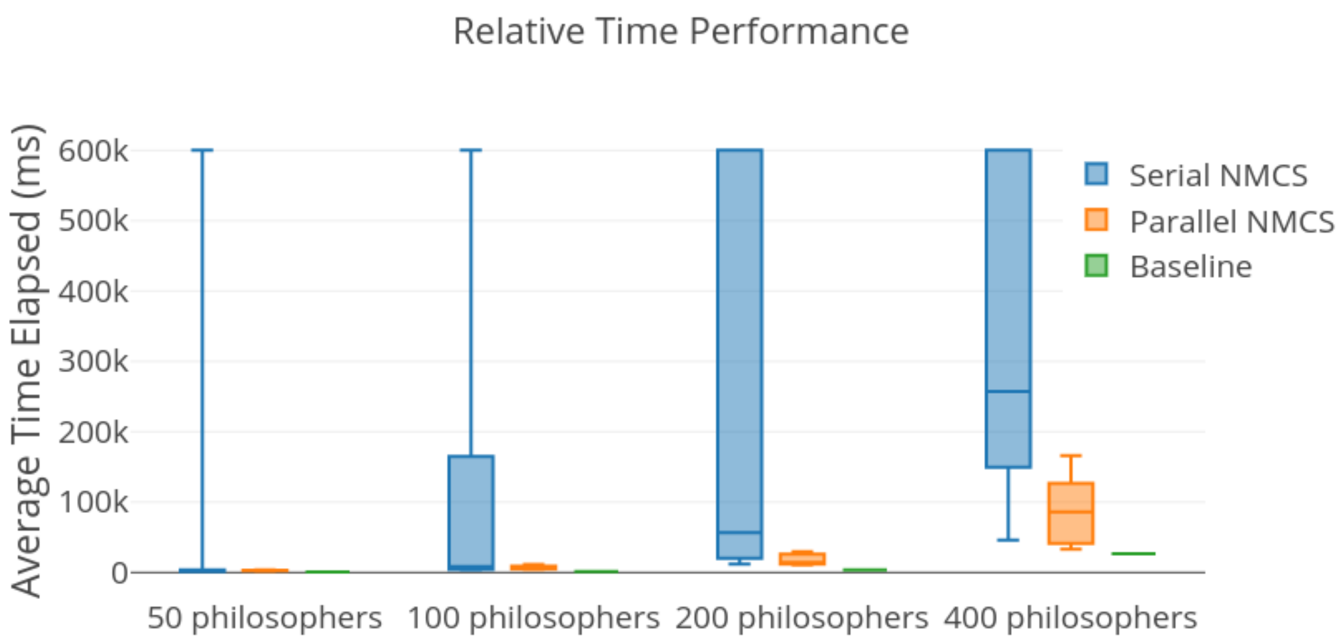
Evaluation: Results





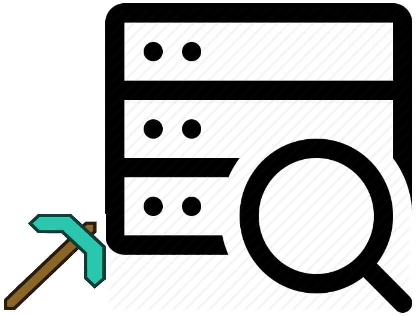
- In terms of the number of states explored (or, alternatively, memory consumption), NMCS offers competitive performance while being prior-free.
- Counterexample lengths are identical (+/- 1) to baseline.

Evaluation: Results

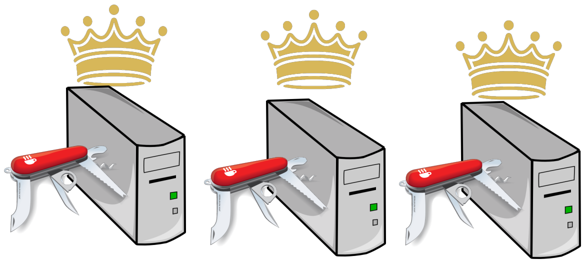


- With 4 workers, parallel NMCS is 22.84x, 22.53x, 16.55x, and 5x faster than the serial NMCS.
- Parallel NMCS is 3-4x slower than the serial baseline. This is an embarrassingly parallel task, so we can go even lower, but turnaround time will still be limited by communication overhead.

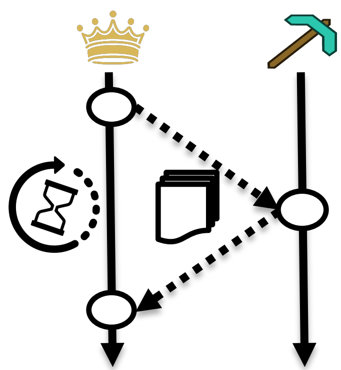
Where do we go from here?



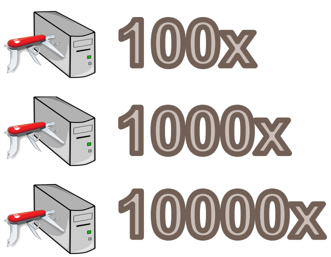
(1) Smarter caching policies for workers can enable better reuse, faster turnaround time.



(2) Search partitioning schemes can be used to enable multiple master instances.



(3) Master/Worker computation can be interleaved to maximize throughput.



(4) Scalability tests against real-world applications of interest will reveal optimization opportunities within JPF.

In Memoriam

Dr. Simon Poulding
October 1967 – August 2017

Questions?