HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

distb

Distributed Model Checking Using ProB

Philipp Körner and Jens Bendisposto

NFM 2018

April 18, 2018

Model Checking Algorithm

Require: formal *specification* and function *desired-property*
  $results := \varnothing$
  $queue := get\text{-}initial\text{-}states(specification)$
  $seen := set(queue)$
  while $queue \neq \varnothing$ **do**
    $state := pop(queue)$
    $invariant\text{-}ok := check\text{-}invariant(specification, state)$
    $successors := compute\text{-}successors(specification, state)$
    $results := results \cup desired\text{-}property(invariant\text{-}ok, successors)$
    for $s \in successors \setminus seen$ **do**
      $enqueue(queue, s)$
    end for
    $seen := seen \cup successors$
  end while
  return *results*

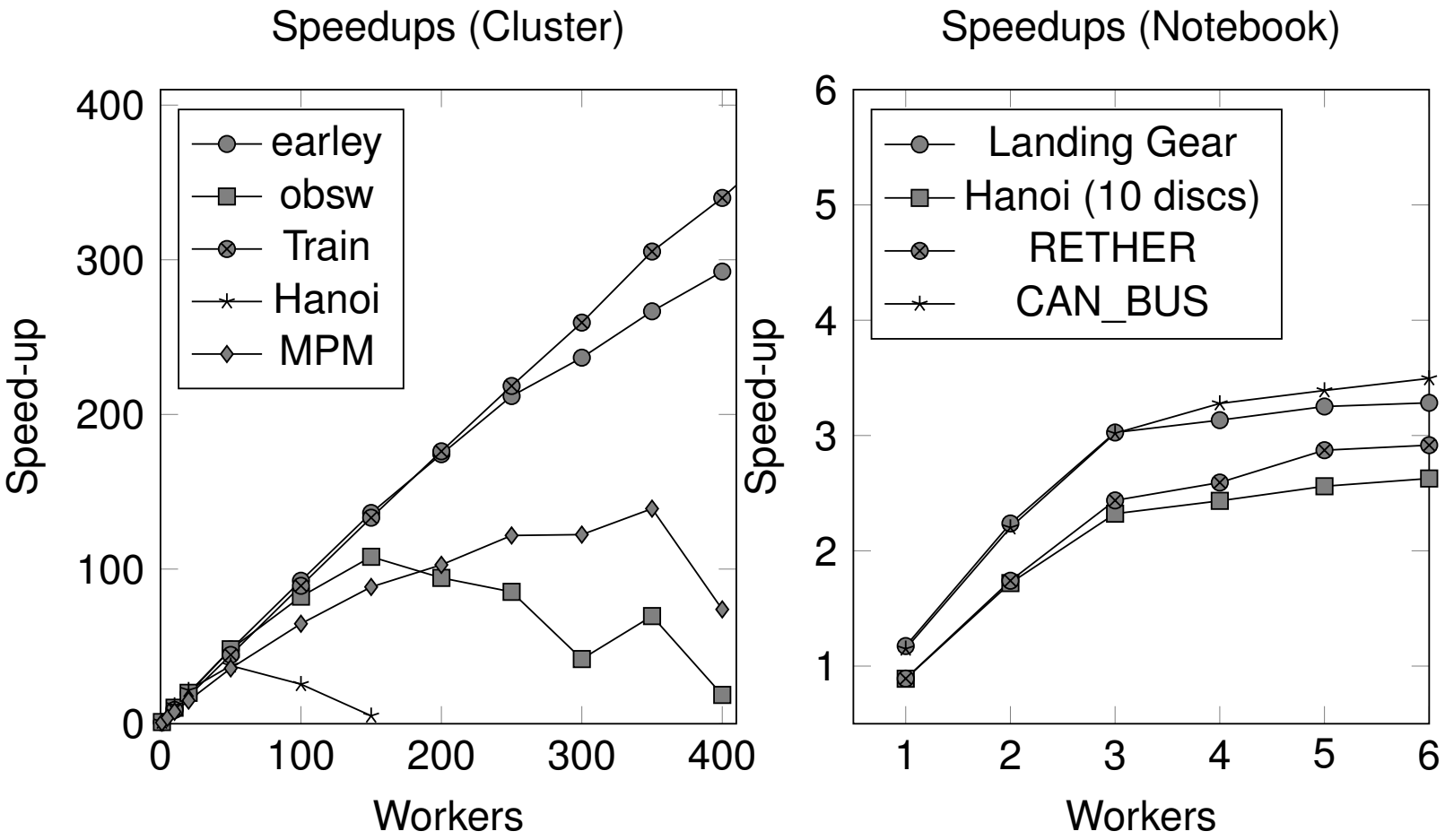
Requirements

- avoid checking the same state multiple times
- avoid network communication when possible
- avoid latency of communication when possible

What is special about B and ProB?

- ProB is implemented in (SICStus) Prolog
- computation of successor states is *slow*
- real models that can be dealt with have few states

Results


Speedups (Cluster)
Speed-up
Workers
earley
obsw
Train
Hanoi
MPM
0
100
200
300
400
Speedups (Notebook)
Speed-up
Workers
Landing Gear
Hanoi (10 discs)
RETHER
CAN_BUS
1
2
3
4
5
6

Suitable Models

- from our experience: everything that is not a counter
- some degree of non-determinism required
- no off-line partitioning of the state space

Architecture

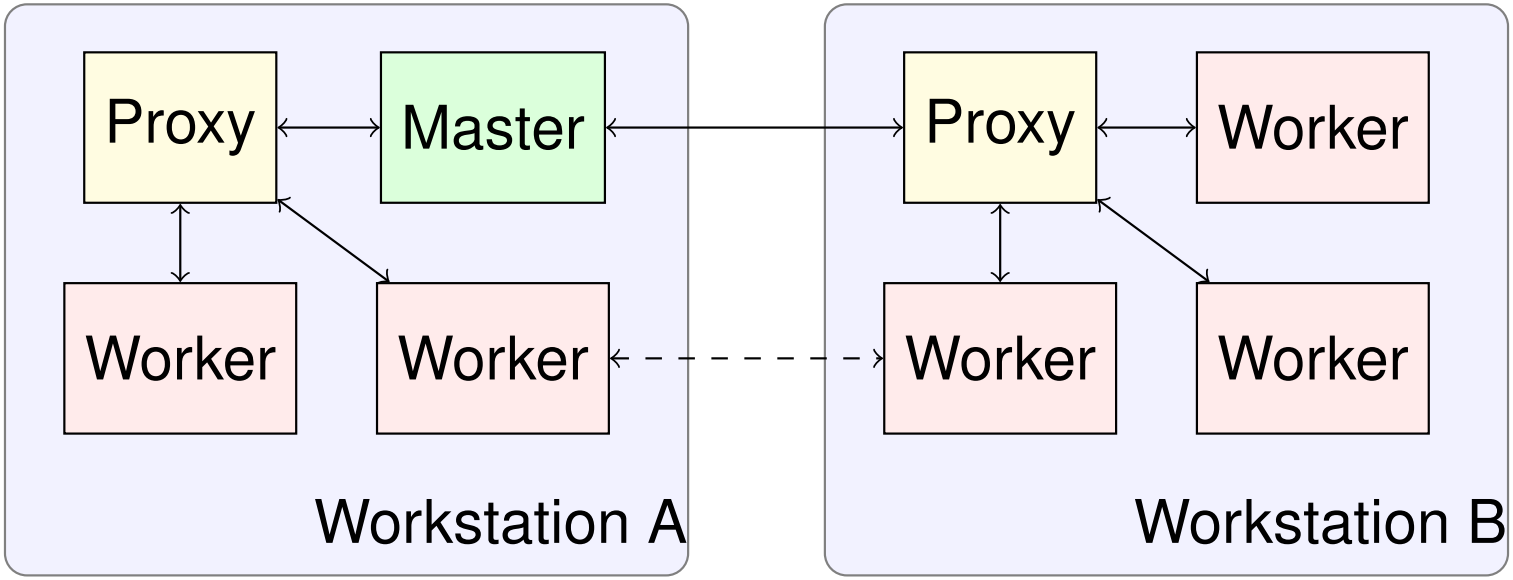


Figure: Typical Setup in a Distributed Setting.

ZeroMQ Zocket Types

Master	Proxy	Worker	Messages and Usage
REP	REQ		ID distribution
	REP	REQ	
PULL	PUSH		hash codes and results
	PULL	PUSH	
PUB	SUB, PUB		hash code propagation, sending commands
	PUB	SUB	sending commands
		REP	receiving work
REQ	REQ	REQ	sending work

Communicating States

- states are Prolog terms
- states have to be serialized in order to . . .
  - . . . send them to other workers
  - . . . get hashed
- undocumented library (fastrw) avoids parsing
- deserialization is still not cheap though

Visited States

- visited states are hashed
- full states are only communicated to share queues
- we assume: same hash $\Longrightarrow$ same state
- default hash function: SHA-1 (160 Bit)

Visited States: Data Structure

Based on Bagwell's Hash Array Mapped Tries

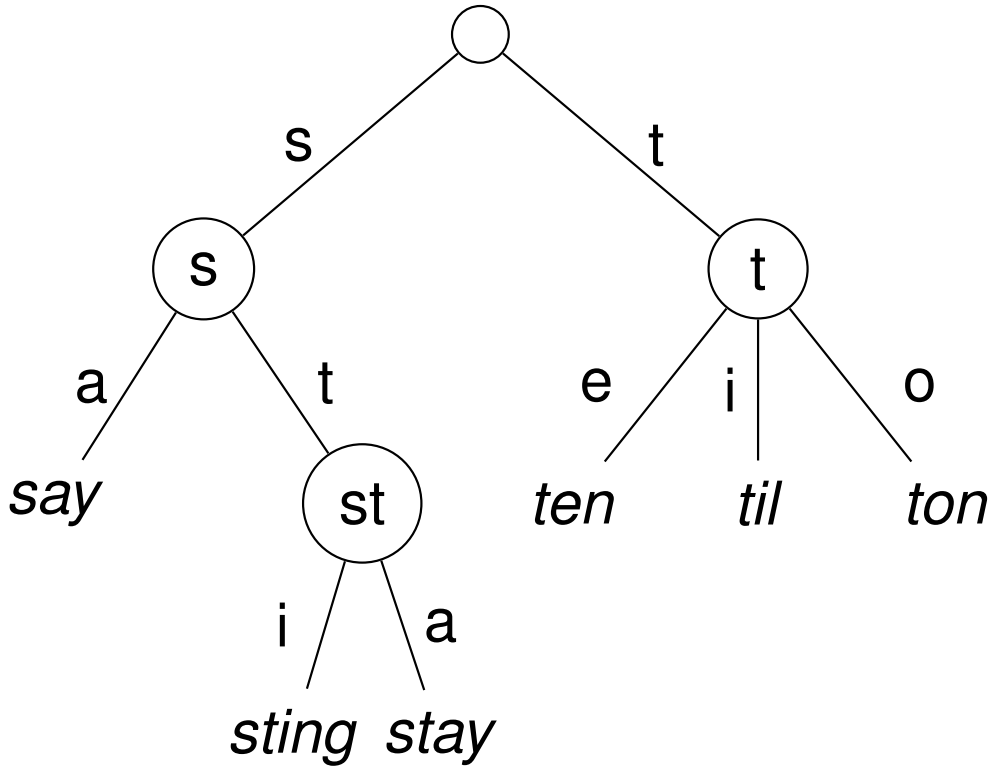


Figure: An Example for an 26-ary[1] Trie

[1] actually 27-ary, 26 letters plus end-of-word symbol

Work Sharing

- obvious solution: every worker reports queue size on change
- use a queue threshold
- proxy communicates: `empty`, `working` and `can_share`
- master initiates transfers between machines
- proxy initiates transfers on the same machine

Proxy

- a hack to avoid file handle per process limits
- a tool to reduce bandwidth consumption
- coordinator of shared memory

Bandwidth Reduction

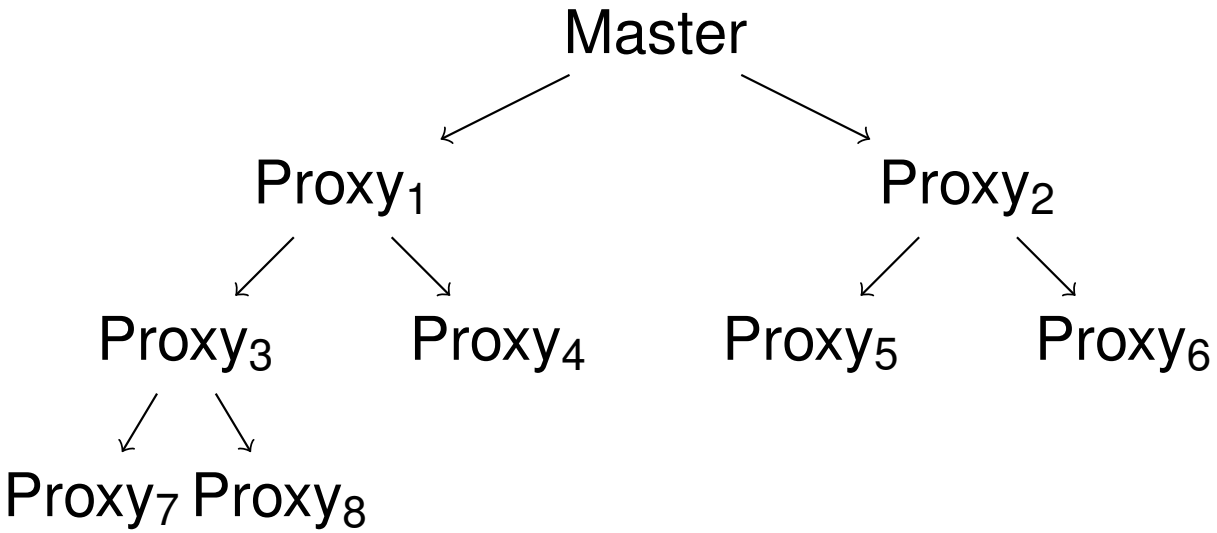


Figure: Hash Code Streaming Tree.

Distributed Model Checking for B

- *distb* (ProB)
- LTSmin (ProB)
- TLC (translation to TLA$^+$)
- . . . ?

Summary

- some interesting techniques are applicable
- distributed MC works well for high-level languages (e.g., B)
- large models that were unfeasible for ProB can be checked over lunch

PS:

- my manuscript for this talk is available:
- `https://github.com/pkoerner/presentations`
- I accept pull request :-)