# State Space Explosion or: How To Fight An Uphill Battle

## A practitioners guide to model checking with TLC

Markus A. Kuppe

Microsoft Engineer

TLA$^+$ Quinceańera - July 18, 2018

# New TLA$^+$ Tools and Toolbox Release

- Tutorial aimed at TLA$^+$ Toolbox 1.5.7 and TLC 2.13 release as of July 18, 2018 (changelog)

# State Space Explosion

Problem of (explicit state) model checking:

*Linear* increase in size of specification or properties can lead up to *exponential* growth of state space

# State Space Explosion

| qspinlock | 1 process | 2 proc | 3 proc | 4 proc[1] |
|---|---|---|---|---|
| Distinct States | 7 | $\sim 10^4$ | $\sim 3.6 \times 10^7$ | $> 1.2 \times 10^{10}$ |
| Run-time | $1s$ | $\sim 30s$ | $\sim 6m$ | Days |

[1] Symmetry & 32 core machine

# TLC

- Safety checking corresponds to Breadth-First search over on-the-fly generated state graph
- Liveness checking corresponds to Depth-First search over (partial) behavior graph [see Manna and Pnueli, 1995, for algorithm]
  - Behavior graph is cross-product of state graph and liveness tableau graph (if any)

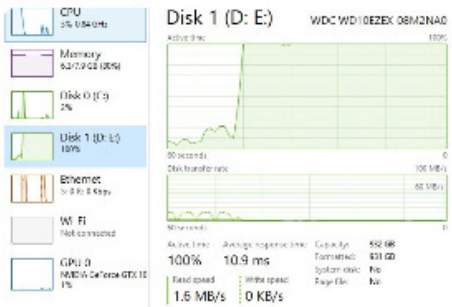
# TLC

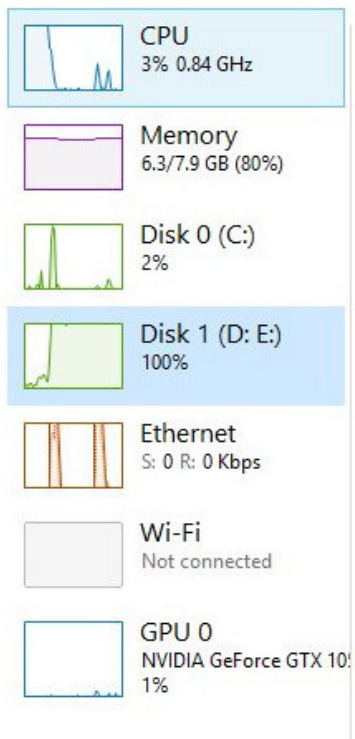
| | |
|---|---|
| FPS | Fingerprint Set<br>(contains a hash of all already explored states) |
| SQ | State Queue<br>(contains all unexplored states) |
| T | Trace<br>(paths from initial states to any state (forest)) |
| WORKER | Set of worker nodes<br>(nodes/cores executing init and next-state relations) |
| $\phi$ | Safety & Liveness properties<br>(as defined by the model) |

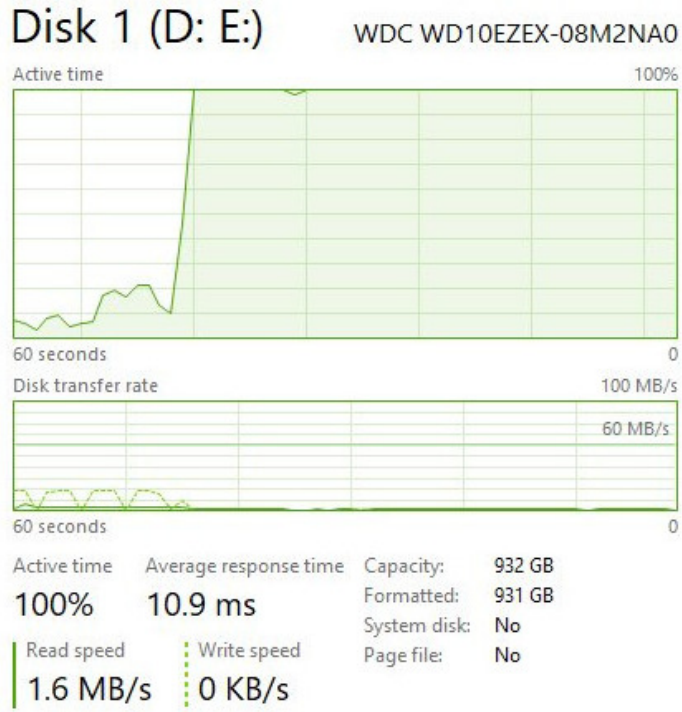
# IO-bound Model Checking

**Andrew Helwer** @ahelwer 8h

lol always a pleasure to hit the TLC I/O-bound death graph

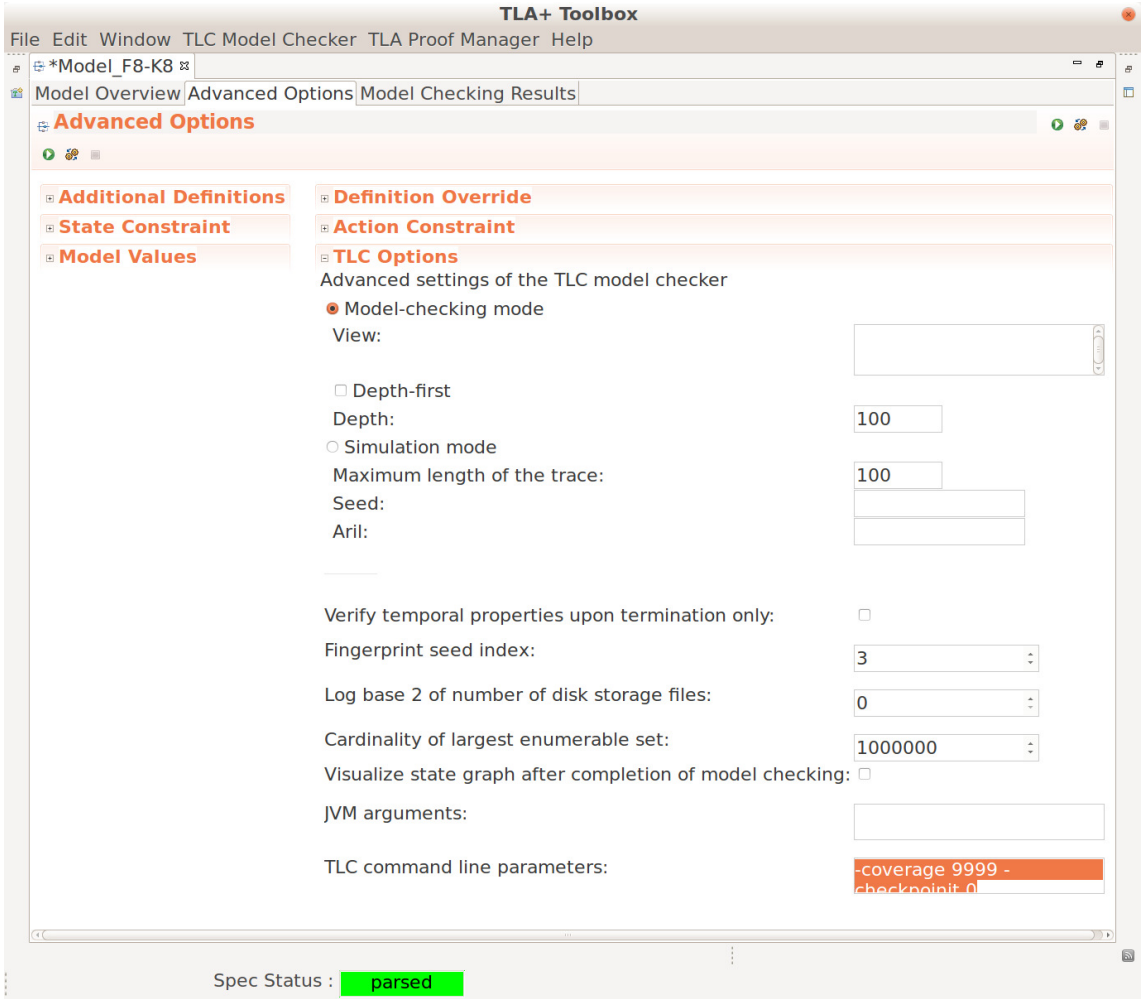


# Low Hanging Fruits

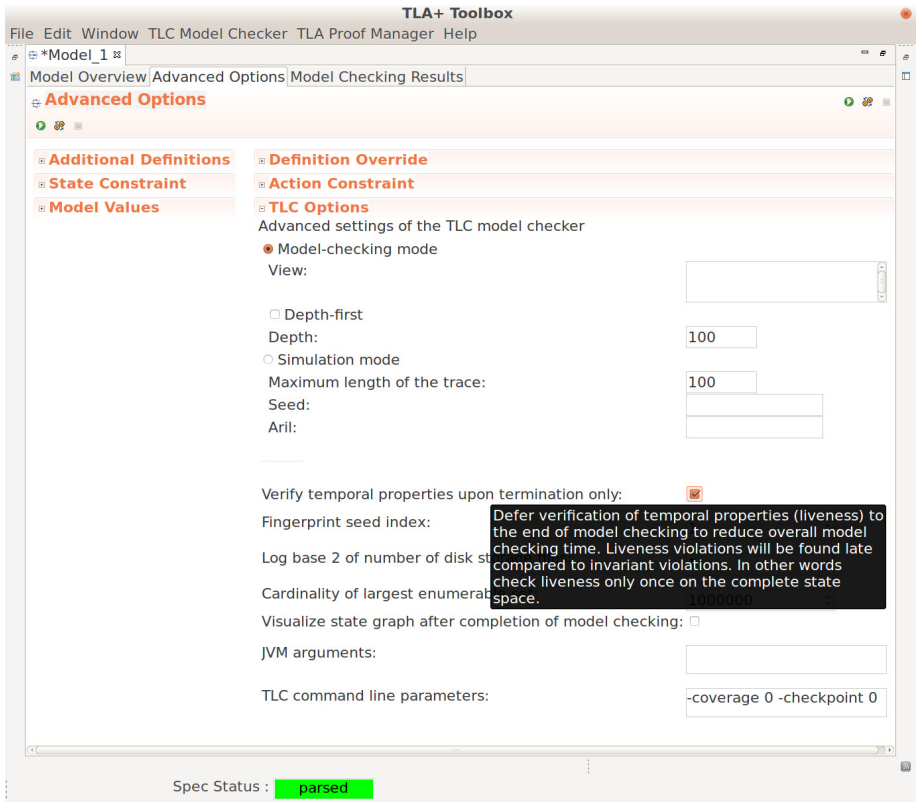
1. Disable checkpoints via "`-checkpoint 0`"[2]
2. Reduce coverage reporting via "`-coverage 99999`"




[2] Manually trigger checkpointing via JMX.

# Postpone Liveness Checking[3]

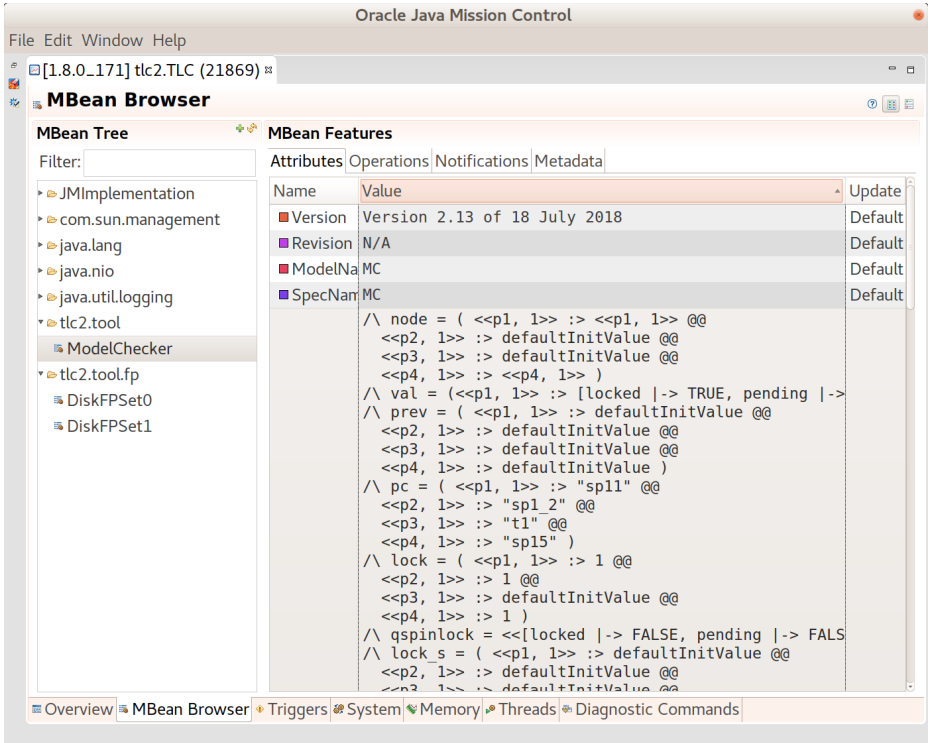
- TLC - by default - periodically checks liveness on partial state graph
- Liveness checking not done incrementally
  - E.g. “periodic” adds 40 mins to a $\sim$ 2h run
- TLC parameter: “-lncheck final”




[3] Manually trigger liveness checking via JMX.

# JMX front-end

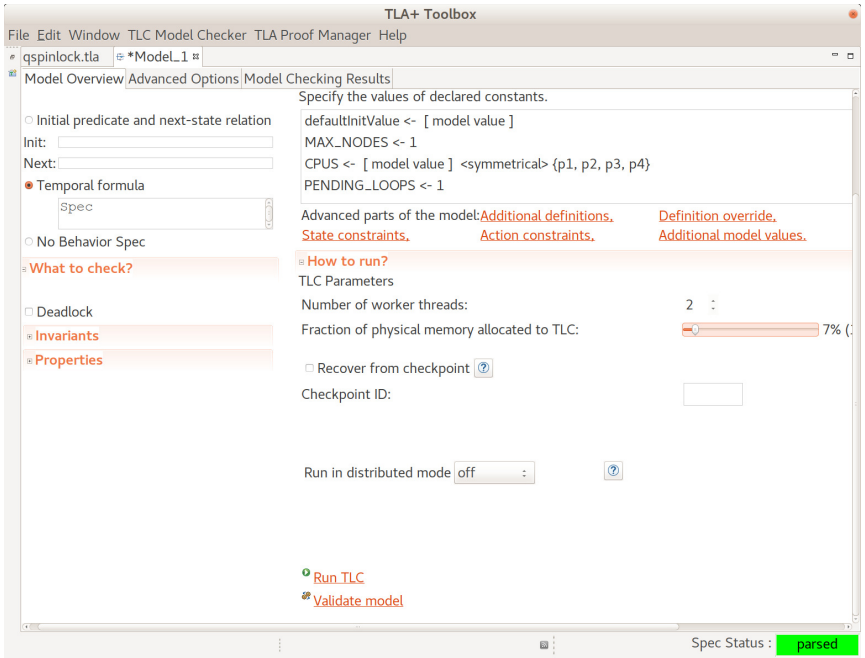
- Connect to local or remote TLC process with e.g. Java Mission Control (included in JDK)
- Large amounts of telemetry data
- Next-State “Inspection Hatch”
- TLC operations
  - Liveness checking
  - Checkpointing



# Demo time

# Parallel TLC

- Number of workers defaults to 1/2 machine's core count
  - because TLC is assumed to run on your workstation
- Drawback
  - No strict Breadth-First search
  - $\sim>$ Approximates shortest error trace

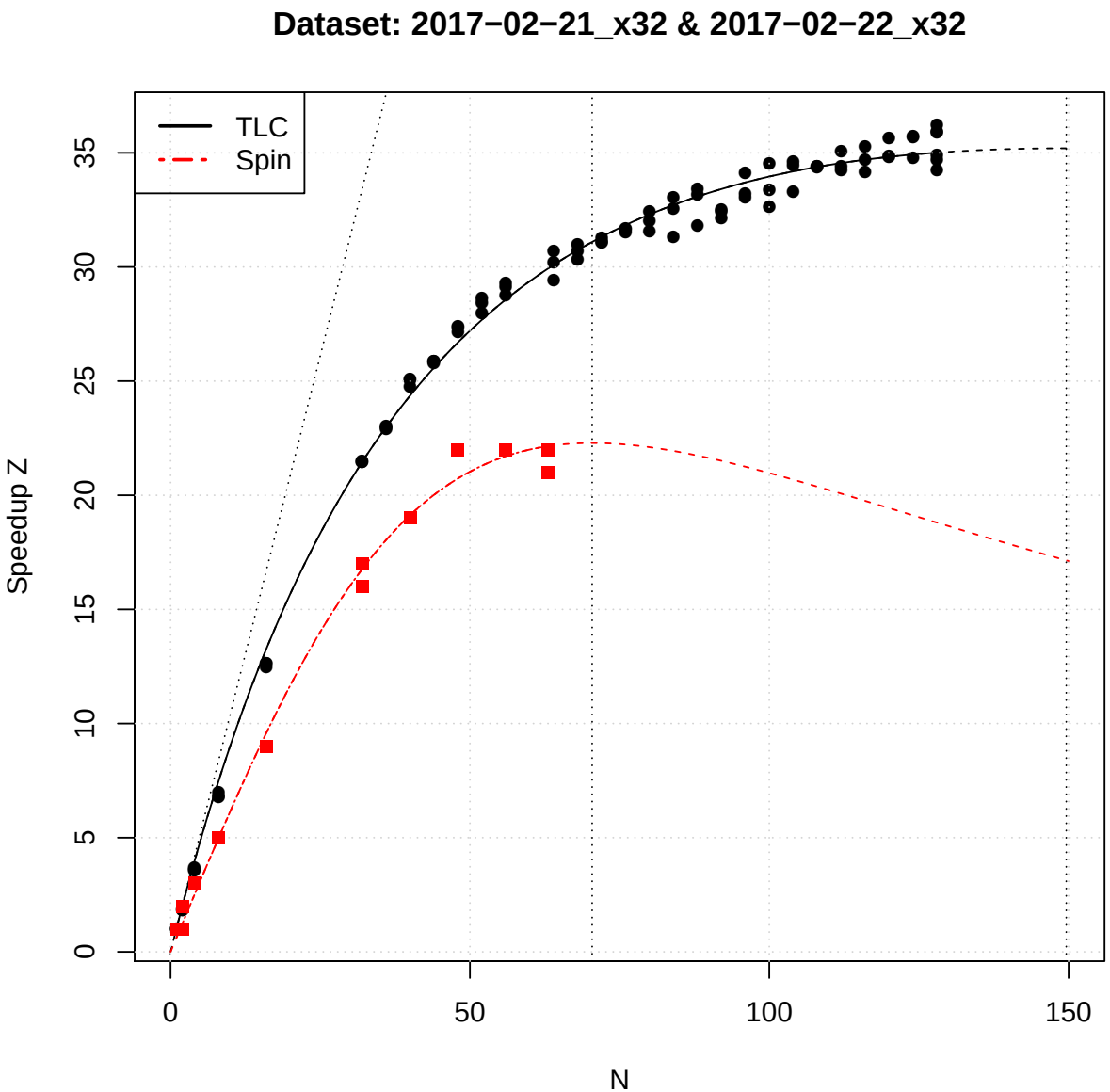


# Parallel TLC

Prototype reduces worker contention via:

1. Partitioned/Sharded FPS
2. Trace T per worker
3. Shared-Nothing State Queue SQ
    3.1 More data to validate underlying assumptions about average shape/properties of state graphs!

# Parallel TLC




- TLC prototype
- TLC beats scalability of SPIN Holzmann [2001]
  - Bakery spec
- (SPIN outperforms throughput of TLC)

# Outline

# Order of Expressions

VARIABLES $x$, $y$

$expensiveOp(n) \triangleq$ CHOOSE $e \in$ SUBSET $(1 \,..\, n) :$ true

$Init \triangleq x = 0 \land y = 0$

---

NextA, NextB or "This is math so who cares!" ?

$$
\begin{aligned}
NextA \triangleq\ & \land x = 0 \land y = 0 \\
& \land x' \in 1 \,..\, 10000 \\
& \land y' = expensiveOp(18)
\end{aligned}
$$

---

$$
\begin{aligned}
NextB \triangleq\ & \land x = 0 \land y = 0 \\
& \land y' = expensiveOp(18) \\
& \land x' \in 1 \,..\, 10000
\end{aligned}
$$

# Order of Expressions

ASSUME $(TLCSet(42, \langle \text{“expensiveOp”}, 0 \rangle)$

$expensiveOp(n) \triangleq$ CHOOSE $e \in$ SUBSET $(1 \,..\, n) : \text{true}$
$\quad \wedge\ TLCSet(42, [TLCGet(42) \text{ except } ![2] = @ + 1])$

$Next \triangleq \wedge \ldots$
$\quad \wedge\ PrintT(TLCGet(42))$

---

NextA:
<<"expensiveOp", 1>>
<<"expensiveOp", 2>>
<<"expensiveOp", 3>>
...

NextB:
<<"expensiveOp", 1>>
<<"expensiveOp", 1>>
<<"expensiveOp", 1>>
...

# Order of Expressions

VARIABLES $x$, $y$

$expensiveOp(n) \triangleq$ CHOOSE $e \in$ SUBSET $(1 \,..\, n) :$ true

$Init \triangleq x = 0 \land y = 0$

---

NextA, NextB or "This is math so who cares!" ?

$$
\begin{aligned}
NextA \triangleq\ & \land x = 0 \land y = 0 \\
& \land x' \in 1 \,..\, 10000 \\
& \land y' = expensiveOp(18)
\end{aligned}
$$

Killed TLC after a few minutes

---

$$
\begin{aligned}
NextB \triangleq\ & \land x = 0 \land y = 0 \\
& \land y' = expensiveOp(18) \\
& \land x' \in 1 \,..\, 10000
\end{aligned}
$$

TLC takes 10 seconds

# Detour: Print & PrintT

Send large volumes of user-generated output (Print and PrintT) to a separate output file with TLC's userFile parameter[4]:

```
-userFile /path/to/file.out
```

[4] Cannot just redirect TLC's stdout because Toolbox relies on it.

# TLCEval?

VARIABLES $x$, $y$

$$expensiveOp(n) \triangleq \text{CHOOSE } e \in \text{SUBSET } (1 \,..\, n) : \text{true}$$

$$Init \triangleq x = 0 \land y = 0$$

$$\begin{aligned} NextA \triangleq\ & \land x = 0 \land y = 0 \\ & \land x' \in 1 \,..\, 10000 \\ & \land y' = TLCEval(expensiveOp(18)) \end{aligned}$$

# TLCEval?

VARIABLES $x$, $y$

$$expensiveOp(n) \triangleq \text{CHOOSE } e \in \text{SUBSET } (1 \,..\, n) : \text{true}$$

$$Init \triangleq x = 0 \land y = 0$$

$$\begin{aligned} NextA \triangleq\ & \land x = 0 \land y = 0 \\ & \land x' \in 1 \,..\, 10000 \\ & \land y' = TLCEval(expensiveOp(18)) \end{aligned}$$

Nope, doesn't work (Eagerly evaluates Sets and FcnRcds only)!

# Constant Folding?

CONSTANT $K$

$$idx(fp,\, p) \triangleq \text{if } \exists\, n \in 1\,..\,K : 2^n = K \text{ then } bitshift(fp,\, p) \text{ else } rescale(K,\, max(fps),\, min(fps),\, fp,\, p)$$

$[\ldots]$

$$Next \triangleq \ldots \wedge \exists\, i \in 1\,..\,10000 : idx(4711,\, i)$$

# Human Constant Folding Only!

CONSTANT $K$

$isKPowerOfTwo \triangleq \exists\, n \in 1\,..\,K : 2^n = K$

$idx(fp,\, p) \triangleq$ if $isKPowerOfTwo$
then $bitshift(fp,\, p)$
else $rescale(K,\, max(fps),\, min(fps),\, fp,\, p)$

$[\ldots]$

$Next \triangleq \ldots \wedge \exists\, i \in 1\,..\,10000 : idx(4711,\, i)$

# Outline

# Module Overwrites

- Implement TLA$^+$ operators in Java
  - Almost all standard modules are overwritten
- Drawbacks
  - Just TLA$^+$ operators
  - Correctness of module overwrites?!
  - Hack to test implementation code

# Module Overwrites

MODULE *ContrivedExample*

$$expensiveOp(n) \triangleq \text{CHOOSE } e \in \text{SUBSET } (1 \,..\, n) : \text{true}$$

```
public class ContrivedExample {
  public static Value expensiveOp(final IntValue n) {
    Random rnd = new Random(EnumerableValue.
        getRandomSeed());

    IntervalValue iv = new IntervalValue(1, n.val);
    int size = iv.size();

    ValueVec vec = new ValueVec();
    for (int i = 0; i < rnd.nextInt(size - 1) + 1; i++)
        {
      vec.addElement(iv.elementAt(rnd.nextInt(size)));
    }
    return new SetEnumValue(vec.sort(true), true);
  }
}
```

# Module Overwrites

- Good to replace Inv with module overwrite

---

CONSTANT *fps*, *empty*
VARIABLES *table*

[...]

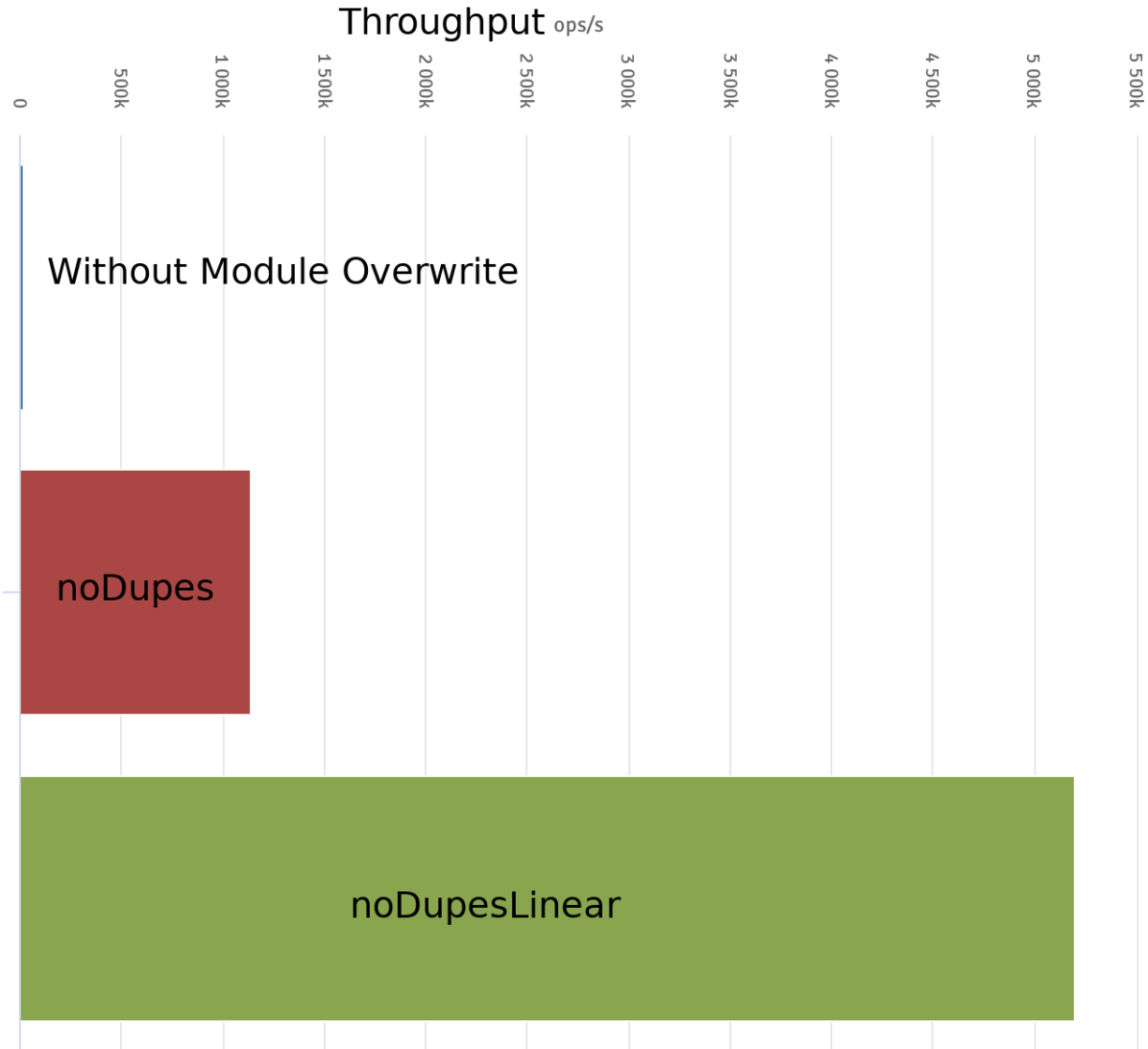
$$noDupes(range, t, emp) \triangleq \text{LET } sub \triangleq SelectSeq(t, \text{lambda } e : e \neq emp)$$
$$\text{IN} \quad \text{if } Len(sub) < 2 \text{ then true}$$
$$\text{else } \forall i \in 1 \,..\, (Len(sub) - 1) :$$
$$\forall j \in (i+1) \,..\, Len(sub) :$$
$$abs(sub[i]) \neq abs(sub[j])$$

$$Inv \triangleq noDupes(fps, table, empty)$$

# Module Overwrites

```
public static Value noDupes(IntervalValue fps,
    FcnRcdValue frv, ModelValue exclude) {
      boolean[] arr = new boolean[fps.size()];
      for (Value value : frv.values) {
            if (value != exclude) {
                  int val = Math.abs(((IntValue) value)
                      .val);
                  if (arr[val - fps.low] == true) {
                        return BoolValue.ValFalse;
                  }
                  arr[val - fps.low] = true;
            }
      }
      return BoolValue.ValTrue;
}
```

# Module Overwrites




(see online)

# Outline

# Distributed TLC

**Andrew Helwer**
@ahelwer

Following

@lemmster perhaps distributed TLC is in my future! 🤔🤔🤔there's no way I'm checking temporal properties on this thing anyway.

5:24 AM - 10 Apr 2018

# Distributed TLC

- Executes TLC on network of machines
- Scales close to linear for "simple" states (serialization)
- Drawbacks
  - Approximates BFS
  - Master (SQ) is SPOF & bottleneck
    - Checkpointing
  - No liveness checking
  - Difficult to setup

# Distributed TLC

- Performance of single-node TLC degrades as soon as fingerprint set goes to disk
  - (Solid state) disks order of magnitude slower compared to RAM
- Even remote memory still faster
- $\sim>$ Use distributed FPS

# Outline

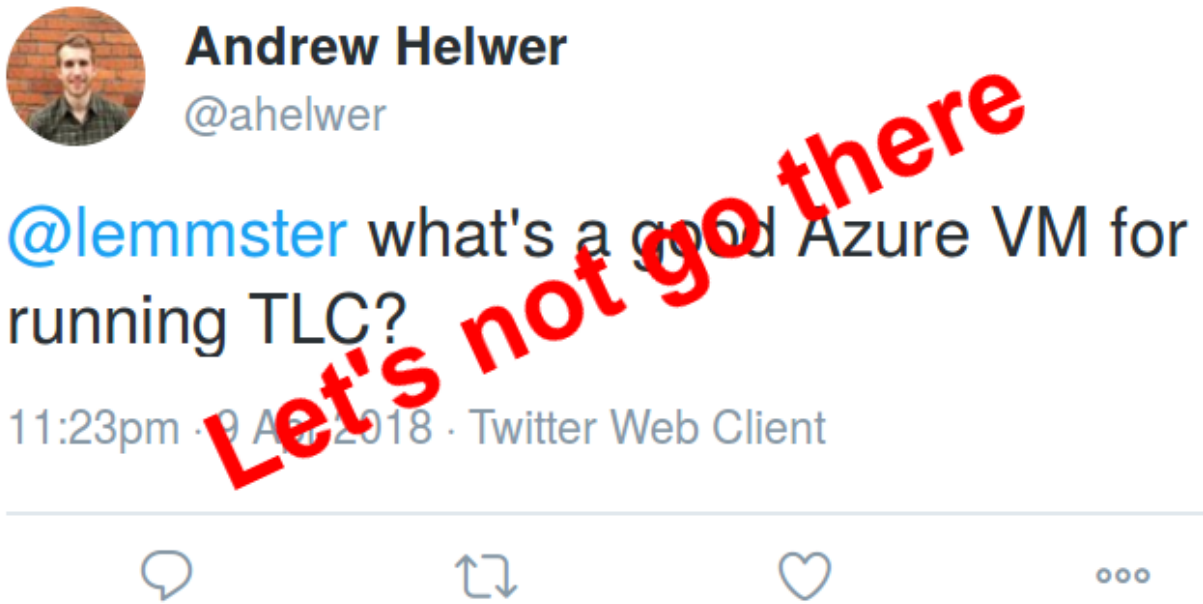
# Cloud TLC

**Andrew Helwer**
@ahelwer

@lemmster what's a good Azure VM for running TLC?

11:23pm · 9 Apr 2018 · Twitter Web Client

# Cloud TLC


Andrew Helwer
@ahelwer
@lemmster what's a good Azure VM for running TLC?
11:23pm · 9 Apr 2018 · Twitter Web Client
Let's not go there

# Cloud TLC

- Push-Button model checking in the cloud
  - Support for Azure and AWS
    - Just compute APIs for portability reasons
  - Hide away idiosyncrasies of TLC and cloud platform
  - Easily check several models concurrently
- Support for single node TLC and Distributed TLC
- Can be started from Toolbox and CLI

# Cloud TLC

- Toolbox release 1.5.7 reduces Cloud TLC start-up time from minutes to seconds
- $\sim>$ Repeatedly run Cloud TLC even on "smallish" models
- Instance count is elastic with regards to resource demand
- Instances dispose automatically after inactivity

# Demo time

# Cloud TLC via CLI

```
## Launch Cloud TLC (here installed to /opt/TLA+
    Toolbox) on the model located at /path/to/model
    / (folder with config and spec)
/opt/TLA+Toolbox/toolbox -nosplash -clean -
    application org.lamport.tla.toolbox.jclouds.
    application /path/to/model/ [aws-ec2|Azure] [
    Model Name] [Spec Name] [Email Address]

## E.g. to Model_1 of specification EWD840:
alias cloudtlc = '/opt/TLA+Toolbox/toolbox -
    nosplash -application org.lamport.tla.toolbox.
    jclouds.application $*'

cloudtlc ~/src/TLA/models/realworld/EWD840/EWD840.
    toolbox/Model_1/ aws-ec2 Model_N9 EWD840
```

# Cloud TLC via CLI

- Automatically creates *Java Flight Recording* (JFR)
  - Almost no overhead
- Downloaded/copied to local machine as tlc.jfr
- Open with Java Mission Control
  - Introspect JVM and TLC

# Outline

# Views

- Explicitly exclude a `VARIABLE aux` from state space exploration
  - "-view" apply VIEW (if provided) when printing out states
  - VIEW on *Advanced Page* evaluated regardless of "-view" parameter
- Drawback
  - Only works for auxiliary variables
- Read *Specifying Systems* page 243

# Outline

# Symmetry Reduction

- Speedup
  - Reduces size of the state space by a factor of up to $|A|!$ where $A$ is the size of the symmetry set
  - Chooses a representative of equivalence classes (orbit) of states
  - BlockingQueue Example
- Drawback
  - Spec or (safety) property might not be symmetric
  - Not supported by liveness checking (TLC prints warning)
  - "Symmetry reduction can have negative effect on TLC performance"
- Read *Specifying Systems* page 245ff

# Symmetry Reduction

MODULE *Symmetry*

EXTENDS *FiniteSets*
VARIABLE $x$
CONSTANT $S$
ASSUME $(Cardinality(S) = 9)$
$Spec \triangleq (x \in S) \wedge \Box[x' \in S]_{\langle x \rangle}$ Without symmetry: 9 states, without: 1

- Let $A$ and $B$ be two symmetry sets:
  - For each state enumerate $|vars| * |A|! * |B|!$ (see TLCStateMut#fingerprint)
- Constructive Orbit Problem - in general - is NP-hard [see Clarke et al., 1998, Donaldson and Miller, 2009]
- (Size of symmetry set $|A|$ limited to 9 elements max unless "*Cardinality of largest enumerable set*" adjusted)

# Symmetry Reduction

MODULE *Symmetry*

EXTENDS *FiniteSets*
VARIABLE $x$
CONSTANT $S$
ASSUME $(Cardinality(S) = 9)$
$Spec \triangleq (x \in S) \wedge \Box[x' \in S]_{\langle x \rangle}$ Without symmetry: 9 states, without: 1

- Let $A$ and $B$ be two symmetry sets:
  - For each state enumerate $|vars| * |A|! * |B|!$ (see TLCStateMut#fingerprint)
- Constructive Orbit Problem - in general - is NP-hard [see Clarke et al., 1998, Donaldson and Miller, 2009]
- (Size of symmetry set $|A|$ limited to 9 elements max unless "*Cardinality of largest enumerable set*" adjusted)

# Outline

# Unfeasibly Large State Spaces

MODULE *Foo*

EXTENDS *Integers*

VARIABLE $x$

$Init \triangleq x \in \text{SUBSET } (1 \,..\, 500)$

$[\ldots]$

- $|SUBSET\ (1..500)| = 10^{150}\ (= 2^{500})$
- *"There are about* $10^{80}$ *atoms in the observable universe, by rough estimation."* (Wikipedia)

# Unfeasibly Large State Spaces

MODULE *Foo*

EXTENDS *Integers*

VARIABLE $x$

$Init \triangleq x \in \text{SUBSET } (1 \,..\, 500)$

$[\ldots]$

- $|SUBSET\ (1..500)| = 10^{150}\ (= 2^{500})$
- "*There are about* $10^{80}$ *atoms in the observable universe, by rough estimation.*" (Wikipedia)

# Unfeasibly Large State Spaces

MODULE *Foo*

EXTENDS *Integers*

VARIABLE $x$

$TypeOK \triangleq x \in \text{SUBSET } (1 \,..\, 500)$

$Init \triangleq x = \{\}$

$Next \triangleq x' = \ldots$

# Trick to find Inductive Invariant candidates with TLC

MODULE *Foo*

EXTENDS *Integers*

VARIABLE *x*

$TypeOK \triangleq x \in \text{SUBSET } (1 \,..\, 500)$
$H \triangleq \ldots$ "interesting part"
$Inv \triangleq TypeOK \wedge H$

$Spec \triangleq Inv \wedge \Box[\ldots]_{\ldots}$ Make Inv the initial predicate.

# Trick to find Inductive Invariant candidates with TLC

MODULE *Foo*

EXTENDS *Integers*

VARIABLE $x$

$TypeOK \triangleq x \in InterestingSetOfSubsets(\text{SUBSET } (1 \,..\, 500))$
$H \triangleq \ldots$
$Inv \triangleq TypeOK \wedge H$

$Spec \triangleq Inv \wedge \Box[\ldots]_{\ldots}$

# New Standard Module Randomization

MODULE *Randomization*

RandomSubset(k, S) equals a randomly chosen subset of S containing k elements, where 0 < k < Cardinality(S).

$$RandomSubset(k, S) \triangleq \text{CHOOSE } T \in \text{SUBSET } S : Cardinality(T) = T$$

RandomSetOfSubsets(k, n, S) equals a pseudo-randomly chosen set of subsets of S – that is, a randomly chosen subset of SUBSET S. Thus, each element T of this set is a subset of S. Each such T is chosen so that each element of S has a probability n / Cardinality(S) of being in T. Thus, the average number of elements in each chosen subset T is n. The set RandomSetOfSubsets(k, n, S) is obtained by making k such choices of T. Because this can produce duplicate choices, the number of elements T in this set may be less than k.

$$RandomSetOfSubsets(k, n, S) \triangleq$$
$$\text{CHOOSE } T \in \text{SUBSET SUBSET } S : Cardinality(T) \leq k$$

# New Standard Module Randomization

MODULE *Foo*

EXTENDS *Integers*, *Randomization*

VARIABLE $x$

$TypeOK \triangleq x \in RandomSubset(4711, \text{SUBSET } (1 \,..\, 500))$
$H \triangleq \ldots$
$Inv \triangleq TypeOK \wedge H$

$Spec \triangleq Inv \wedge \Box[\ldots]_{\ldots}$

- Read "Using TLC to Check Inductive Invariance" (Lamport 2018) which accompanies the new release
- Please provide feedback if this approach is useful when proofing with TLAPS

# Conclusion

- TLC highly tuned as is, difficult to squeeze out more performance :-)
- Reduce constant overhead
  - No coverage, No checkpoints, Final liveness
- Reduce cost of next-state evaluation
  - Order of expressions, Module overwrites
- Provide sufficient memory to stay CPU bound
- Scale TLC vertically (Parallel TLC) and horizontally (Distributed TLC)
- Cloud TLC simplifies running TLC optimized
- Don't shoot yourself in the foot with *symmetry* and *views*

# Outlook

- Future development will continue to focus on scalability of parallel and distributed TLC
  - Please share you specification and *flight recordings* with us
- Parv Mor (GSoC student) is working on a concurrent SCC implementation
- Blue Sky:
  - Truffle/Graal to speed-up evaluation of next-state relation
  - ML to tune/optimize TLC over multiple runs
    - Per spec, per Toolbox or globally
    - User data reporting

*Q&A*

# Contact


- email: makuppe@microsoft.com

# Bibliography I

E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla. Symmetry Reductions in Model Checking. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Alan J. Hu, and Moshe Y. Vardi, editors, *Computer Aided Verification*, volume 1427, pages 147–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-64608-2 978-3-540-69339-0. URL http://link.springer.com/10.1007/BFb0028741.

Alastair F. Donaldson and Alice Miller. On the constructive orbit problem. *Annals of Mathematics and Artificial Intelligence*, 57 (1):1–35, September 2009. ISSN 1012-2443, 1573-7470. doi: 10.1007/s10472-009-9171-4. URL http://link.springer.com/10.1007/s10472-009-9171-4.

# Bibliography II

Gerard J. Holzmann. ACM Software System Award 2001, 2001. URL `http://awards.acm.org/award_winners/holzmann_1625680.cfm`. (Accessed 2017-03-27).

Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, New York, 1995. ISBN 978-0-387-94459-3.