Distributed TLC

Markus A. Kuppe

TLA^{+} Community Event, ABZ 2014 Toulouse, June 3, 2014

Outline

Outline

Highlevel TLC

FPS Fingerprint Set
(contains a hash of all already explored states)

SQ State Queue
(contains all unexplored states)

WORKER Set of worker nodes
(nodes/cores doing heavy calculations)

ϕ Safety properties
(as defined by the model)

Highlevel TLC

FPS Fingerprint Set
(contains a hash of all already explored states)

SQ State Queue
(contains all unexplored states)

WORKER Set of worker nodes
(nodes/cores doing heavy calculations)

ϕ Safety properties
(as defined by the model)

Highlevel TLC

FPS	Fingerprint Set (contains a hash of all already explored states)
SQ	State Queue (contains all unexplored states)
WORKER	Set of worker nodes (nodes/cores doing heavy calculations)
ϕ	Safety properties (as defined by the model)

Highlevel TLC

FPS Fingerprint Set
(contains a hash of all already explored states)

SQ State Queue
(contains all unexplored states)

WORKER Set of worker nodes
(nodes/cores doing heavy calculations)

ϕ Safety properties
(as defined by the model)

Non-distributed TLC

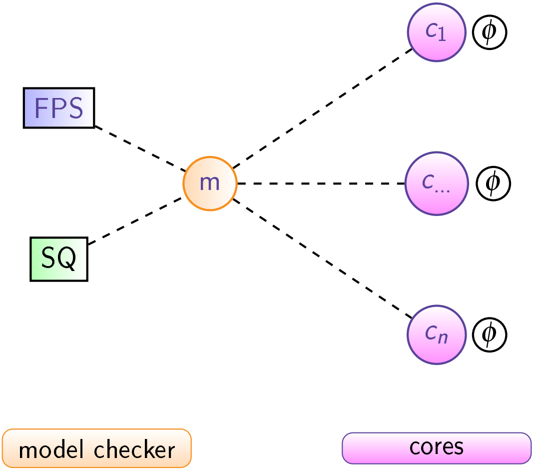

Figure : Non-distributed TLC

Distributed Computation

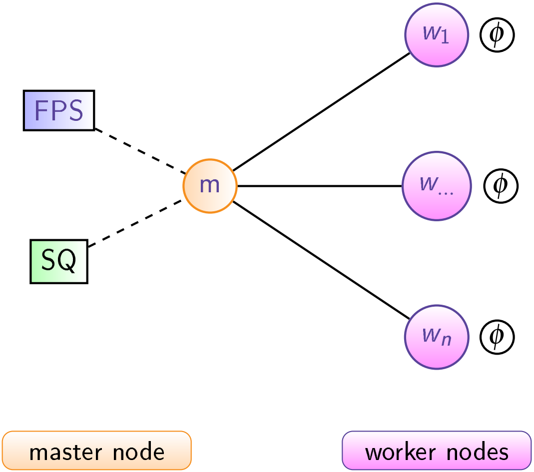

Figure : Compute nodes

Distributed Computation & Storage

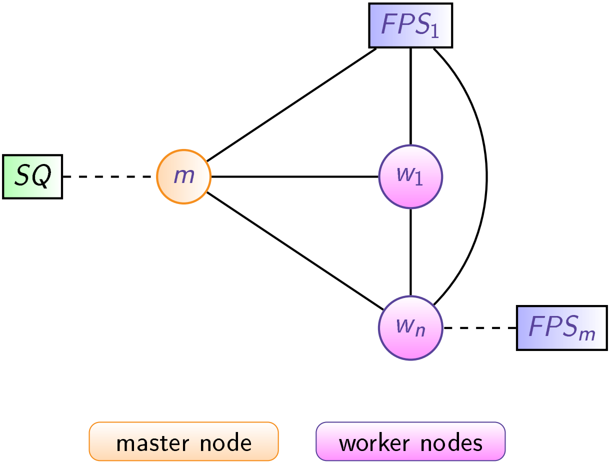

Figure : Compute and storage nodes

When and why?

Compute Use as many cores as possible
(to distribute heavy next-state computation)

Compute & Storage Additionally use as much memory as available
(to speed up FPS lookup)

Scalability

- Scales (almost) linearly
- => Hash collisions

Limitations

- No liveness checking
- No coverage
- No distributed calculation of init states
- No (strict) breadth-first search

Fault tolerance

- 1...n workers (*w*)
- 1...(m-1) fingerprint sets (*FPS*)
- Compensates neither loosing *SQ* nor *Trace*

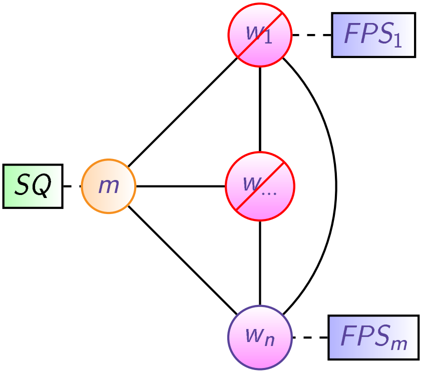

Figure : Broken setup

Outline

Use case

- Experiments block/slow down your desktop

Demo

Demo (Ad-hoc network)

Demo notes (slide not part of final pdf)

1. Create and setup a model for distributed TLC
2. Start model checker (...waiting for workers)
3. Switch to Windows VMs
4. Open IE on Toolbox webserver
5. (Install Java7 via Webstart)
6. Launch Worker in IE in Windows VM
7. Switch to another VM & start more workers

Demo notes (slide not part of final pdf)

1. Start another model
2. Start Distributed FPSet node in VM1
    2.1 Set number of distributed FPSets
3. Start Worker node in VM2

Demo notes (slide not part of final pdf)

1. Start daemon style Worker node in VM first
2. Start small model that completes in less than a minute
3. Start small model that completes ...
4. ...
5. Mention to use Java8 or OOM due to PermGen `http://stackoverflow.com/a/835269`

Common problems

- Broken name resolution
- Blocked by firewalls
- Restrictive system security

Fingerprint Set implementations

- Least Significant Bit (LSB)
  - halves available FPS memory
  - old default
- Most Significant Bit (MSB)
  - exploits all available memory
  - new default
- Off Heap
  - fastest & most efficient
  - Not universally available

Fingerprint Set implementations

- Least Significant Bit (LSB)
  - halves available FPS memory
  - old default
- Most Significant Bit (MSB)
  - exploits all available memory
  - new default
- Off Heap
  - fastest & most efficient
  - Not universally available

Fingerprint Set implementations

- Least Significant Bit (LSB)
  - halves available FPS memory
  - old default
- Most Significant Bit (MSB)
  - exploits all available memory
  - new default
- Off Heap
  - fastest & most efficient
  - Not universally available

Distributed Fingerprint Sets

- Use distributed FPS
  - Performance degrades as soon as TLC goes to disk
    - (Solid state) disks order of magnitude slower compared to RAM
  - Even remote memory still faster

Minor

- Checkpoints
- StateQueue buffer
- Worker cache
- BlockSelector
- ...

Conclusion

- Only simple if it works
- Too many screws

Outline

Use case

- Experiments running many hours to days
- Personal machine not always on

Demo

Demo (in the cloud)

Demo notes (slide not part of final pdf)

- Set AWS credentials as environment variables
- Launch toolbox
- Create and setup model to start on AWS
  - Set drop down to "aws-ec2"
  - Set email
- Mention cloud-specific tuning hard-wired into the Toolbox
- Launch TLC in the cloud
- Show status in browser (have a backup screenshot available)
- "Wait" for email response (have a backup email result ready!)
- Open E-Mail response (result) in Toolbox

Conclusion

- Don't have to worry about systems idiosyncrasies, but you literally pay the price for it

Outline

Summary

- Exploits remote compute power and storage
- Scales
- Ad-hoc deployment can be tricky
- Cloud deployment despite costs way forward

Outlook

- Multi-node cloud deployments
- More pre-sets for cloud instance types
- Support other cloud providers
- “Auto Scaling” based on actual machine load
- Larger (128 bit) fingerprints?

Q&A

Thank you for your attention

Acknowledgment

- Microsoft Research & INRIA Joint Lab
- Amazon AWS
- Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies - https://www.grid5000.fr

References

- eMail: mailto:tla-workshop-2014@lemmster.de
- Slides: https://www.lemmster.de/uploads/DistributedTLC_MarkusAKuppe.pdf