

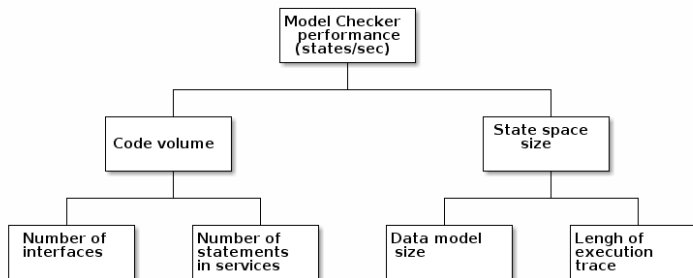
Sbuilder Benchmarks

jj

2017-03-08

TIME TO RUN TLA+TOOLS MODEL CHECKER

- ▶ State space size (states) / Model Checker Performance (states/sec)
- ▶ These benchmarks focus in "performance", for "state space size" refer to Heuristics Used in Creating Runnable Formal Models
- ▶ Scale Model Checker Performance (conclusions spoiler :)
 - ▶ model check large application in chunks
 - ▶ use more workers in model checker (interface binding)
 - ▶ run model checker on distributed hardware (AWS)
- ▶ Performance (states/sec) influenced by code volume and state space:



BENCHMARK APPLICATION & BENCHMARK PARAMETERS

CONTRACT:

- ▶ add interface -> code volume
- ▶ executed -> state space (~ 3 states + statements)

STMT:

- ▶ add statement -> code volume
- ▶ executed -> state space (~ 1 state)

DOMAIN:

- ▶ add domain range -> larger data model -> larger state space
- ▶ no increase in code volume
- ▶ possibility for distribute job to workers!!

ITERATION

- ▶ invoke interface w. bind data -> control state space growth
- ▶ no increase in code volume
- ▶ simulate "dead code" (with invariant ITERATION * CONTRACT)

Simple Ethereum contract (as a Mustache template)

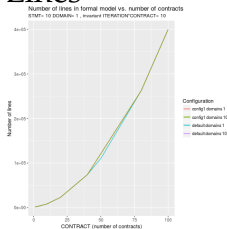
```
// Iterate CONTRACT -times
{{#CONTRACT}}
// Create unique contract
Contract Benchmark_{{contract}} {
    struct s {
        int intVal;
    }

    // Persistent state
    s memberVariable;

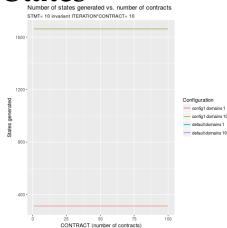
    // Constructor
    Benchmark_contract( string inp1 ) {
        // Iterate STMT times
        {{#STMT}}
        memberVariable.intVal = {{i}};
        {{#STMT}}
    }
}
{{/CONTRACT}}
```

IMPACT OF ADDING INTERFACES

Lines

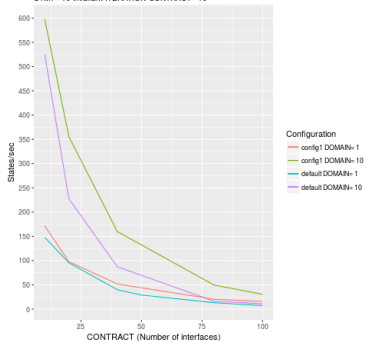


States



Model checking performance on number of interfaces

STMT= 10 invariant ITERATION*CONTRACT= 10



Benchmark runs:

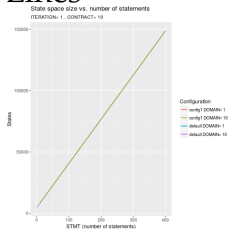
- ▶ vary number of interface, each interface 10 statements, call exactly 10 interfaces (rest "dead code"), configuration: 8/1 workers; 10/1 DOMAINS
- ▶ Number states (for a DOMAIN value) fixed, LOC increase with more interfaces

Results:

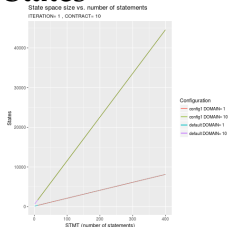
- ▶ performance slopes down from @600 states/sec
- ▶ benefits for more workers & larger data model size

IMPACT OF ADDING STATEMENTS

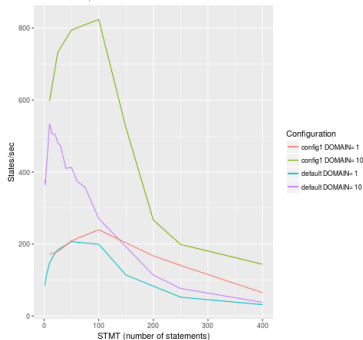
Lines



States



Model checking performance vs. number of statements
ITERATION= 1 , CONTRACT= 10



Benchmark runs:

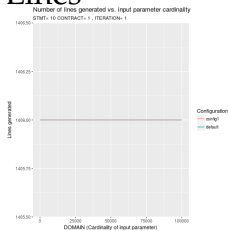
- ▶ vary STMT in interfaces, use 10 interfaces, configuration: 8/1 workers, 10/1 DOMAINS
- ▶ LOC & state grow linearly with STMT, slope of state growth steeper with DOMAIN = 10

Results:

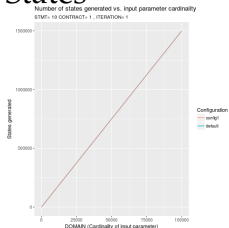
- ▶ performance peaks (@800 states/sec for 10 contracts with 100 statements ~ 1000 statements in application), steep slope down
- ▶ clear benefits when workers can be distributed

IMPACT OF INCREASING DATA MODEL SIZE

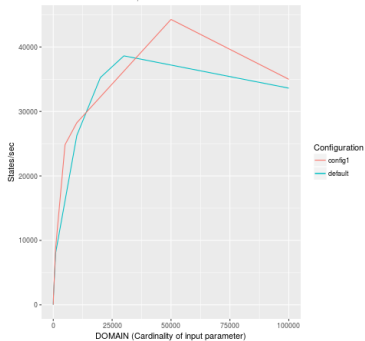
Lines



States



Performance (states/sec) vs. Input parameter cardinality
STMT= 10 CONTRACT= 1 , ITERATIONS= 1



Benchmark runs:

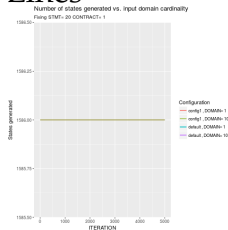
- ▶ one call to one interface, vary data model size (DOMAIN cardinality), with 8 workers/1 worker
- ▶ fixed LOC, state space size grows linearly with DOMAIN param from 0 to > 150.000 states

Results:

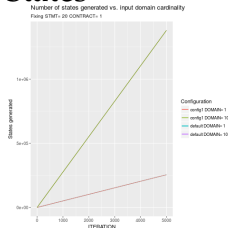
- ▶ quite impressive state & domain sizes
- ▶ peaks @44.000 states/sec for 50.000 domains, gradual slope in performance
- ▶ benefits for more workers

IMPACT OF LONGER EXECUTION TRACES

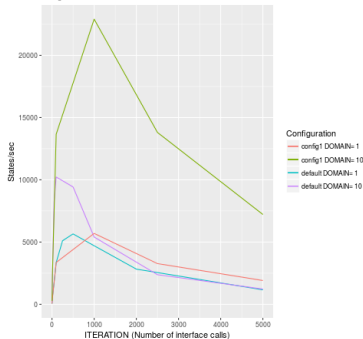
Lines



States



Performance (states/sec) vs. number of interface calls
Fixing STMT= 20 CONTRACT= 1



Benchmark runs:

- ▶ vary number interface invocations (1 interface w. 20 stmts), configuration 8/1 workers, 1/10 domains
- ▶ lines of code: same all configurations
- ▶ states: linear on ITERATION, slope $f(\text{DOMAIN})$

Results:

- ▶ clear benefits from multiple workers, especially when data model cardinality allows splitting job to workers

CONCLUSIONS

- ▶ Running benchmarks on hardware
 - ▶ HP ENVY 17 - Notebook PC
 - ▶ memory 12GB
 - ▶ CPU: Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz, 4 Cores of 8 Threads
- ▶ TLA+tool model checker CPU bound (at least for the benchmarks)
- ▶ Model checking performance (states/sec) slows down severely on formal model size
 - ▶ Can maintain performance > 100 states/sec for models with 60 interfaces (when formal model supports scaling using multiple model checker workers)
 - ▶ 1 state ~ 1 statement in application → 1 minute model checking run ~ execution traces of 6000 statements
- ▶ Scale to real world (large) applications
 - ▶ Split model checking in chunks (in Use Cases): Sbuilder prunes unreachable out code from the formal model of a setup → smaller code volume to model check → faster model check (states/sec)
 - ▶ Clearly benefits from multiple Core w. multiple thread CPU
 - ▶ TLA+tools model checker can be run in distributed mode (on AWS infra), (TBD: run benchmarks using distributed TLA model checker)