Limits to Growth?

Gerard J. Holzmann

1/2/08

64GB
4GB
64KB
16 bit 32 bit 64 bit

RAM size

Clockspeed

4 MHz
4 GHz

1981 2001 2021

“awakening” “golden age” “new age” you are here!

evolution of the desktop PC

10^6 states per second; 10^5 states per second; 10^4 states per second
(up to 10^10 states in bitstate mode, up to 10^8 states in normal mode)

60 60 60 60 60 60 60 60 60 60
10^6
10^7
10^8
10^9
10^10

Seconds

small states
medium coverage
coverage
large states
low coverage

time to fill 10 GB of RAM

scaling

• in one day we can explore no more than 10^10 states
  – no matter how much RAM or disk we have!
  – if Moore’s curve for clock-speed had continued, we
    would have continued to expand this range
• using multi-core, in the best case we could
  increase our range by using multiple CPU cores
  – but concurrency != scalability
• some verification problems are larger than what we
  can handle (and not amenable to symbolic
  methods or abstraction)
  – how do we handle those?
  – the infinite state space and the infinite memory

spin version 5

• supports multi-core verification
• developed on a dual quad-core system with 32 GB of memory
• linear scaling is achieved in the best cases measured

multi-core model checking

leader election (N=9), reference model, and a phone switch model

0
20
40
60
80
100
120

% of Single Core Runtime

leader (N=9)
reference model
spin model

0.5 M states

number of cores
beyond 8 cores: and apparent anomaly

what happened to the nice linear scaling?

hypothesis: are memory caching protocols getting in our way?

memory access on SGI Altix with fast NUMA interconnect

hardware

Altix C-Brick
4 CPUs
(2x dualcore)
2 NUMA links

up to 16 GB per C-brick

Altix R-Brick
4 CPUs
(2x dualcore)
2 NUMA links

64 GB NUMA interconnect (non-uniform memory access)

measurement on the SGI Altix
200,000 states stored, 100 bytes/state

a simple experiment

• a small test program that writes S "states" of V bytes each into memory
  – the program simulates the actions of a model checker: randomly generating states, computing hashes, and storing the state in memory

• execute this program as N parallel threads, with each thread
  1. using separate memory arenas -- comparable to running the threads sequentially
  2. using a shared memory arena with locking

what this means...

• there is a growing performance gap
  – memory size continues to grow
  – but cpu speed no longer does
  – the standard approach to handling large problem sizes has stopped working
    • new algorithms, approaches are needed to leverage large multi-core systems
    • exploiting multi-core systems with shared memory is much harder than it would seem