Tackling Runtime Variance on NUMA Architecture

April 5th, 2012

Agenda

Problem Statement

- Background and Challenges
- Different types of Multi-Processors
- How Caching affects repeatability

Methodology

Sample Results

Minimizing variability (other factors)

Background and Challenges

- **Synopsys Customers' pressure points**
  - Larger designs, shorter design time:
    - EDA application TAT (Turn Around Time) / Performance

- **Synopsys product teams' pressure points**
  - Performance improvement (scalar, Parallel Processing)
    - No runtime performance impact from new features and bug fixes

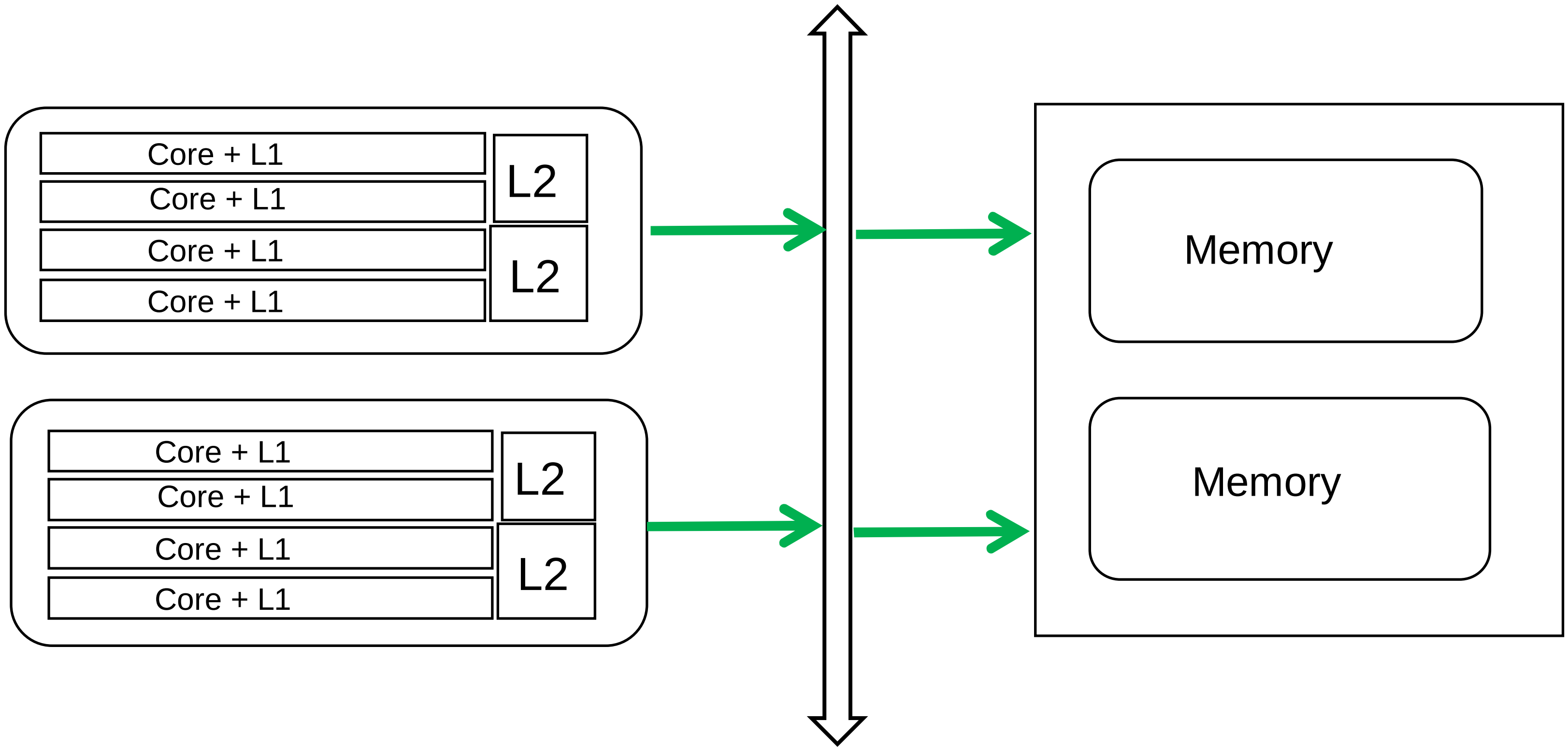
- Greater need for **reliable/consistent performance benchmarking**

Types of Multi-Processors

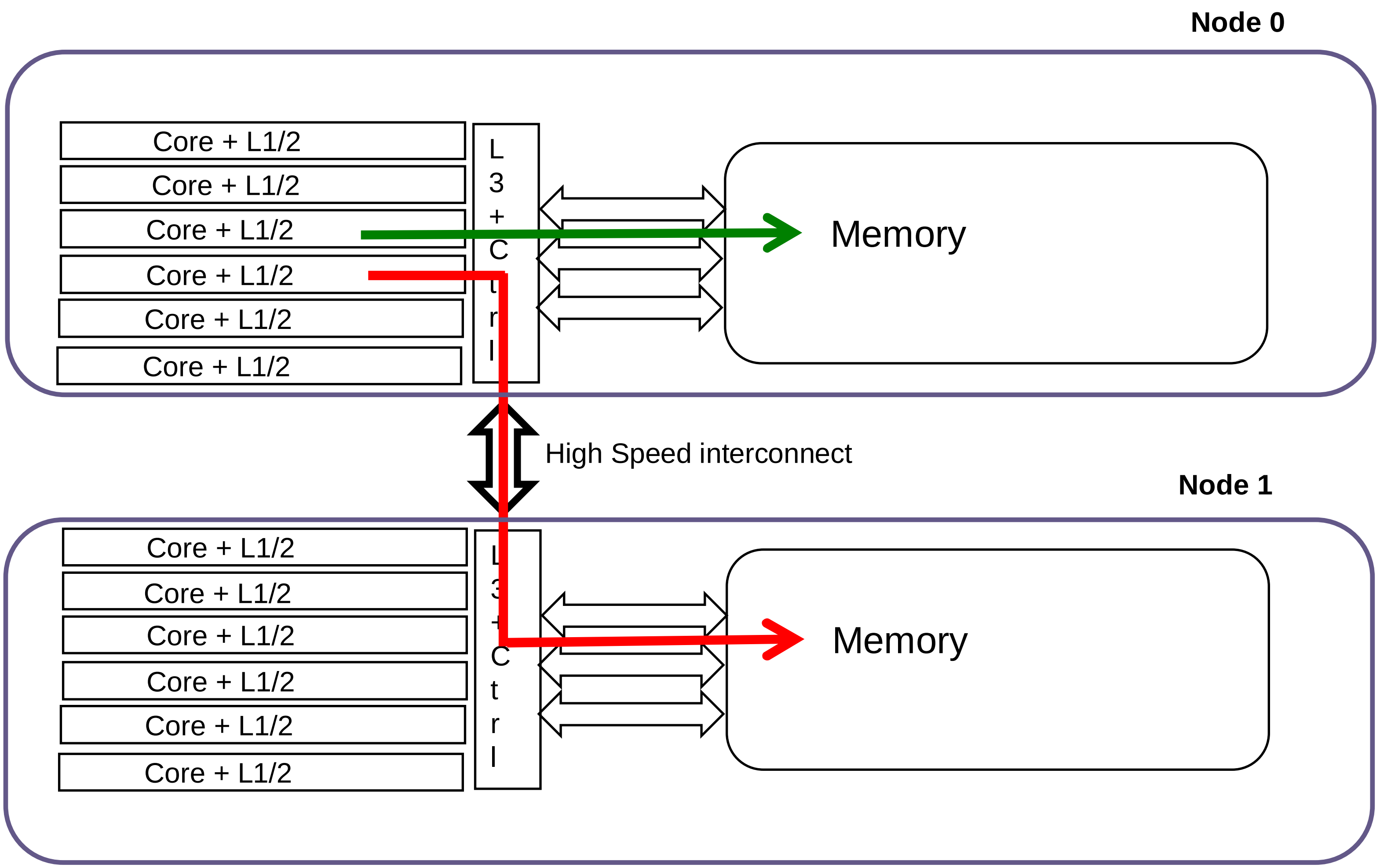
- Architectures based on how memory is accessed
  - SMP -> Symmetric Multi-Processing (Uniform Memory Access)
  - NUMA -> Non-Uniform Memory Access

SMP Architecture

aka UMA



NUMA Architecture



Problem Statement

- How to run Benchmark tests with accuracy and repeatability on NUMA machines?
  - Aiming for 1-3% variation without multiple runs for statistical averaging

How Caching Affects Repeatability
Adds variance

- Cache hits and misses affects system performance
- Cache coloring/Page coloring helps reduce conflict misses
- Linux does not support page coloring
  - True for all Linux variants (RedHat, SUSE)
  - Peak performance is higher, but variability also higher
  - Less deterministic with regards to cache performance

Agenda

Problem Statement

Methodology

- **NUMA hardware settings**
- Load Share setup
- Measuring performance numbers

Sample Results

Minimizing variability (other factors)

NUMA Hardware settings

OS Setup

- **Hardware BIOS settings**

BIOS Setting	Enabled	Description
Turbo Boost	N	Dynamically shutdown unused cores to divert power to over-clock active cores
Hardware PreFetch	Y	Pre-fetch data and instructions from memory to L2 cache. Reduces memory read latency
Hyper-Threading	**N**	**Provides 2 instruction threads per core**

NUMA Hardware settings
Binning

- Process of identifying similar performance machines

- All NUMA machines binned
  - Identical performance machines grouped together
  - Users run Benchmark tests on selected group

Agenda

Problem Statement

Methodology

- NUMA hardware settings
- Load Share setup
  - Making Applications NUMA aware
  - Implementing NUMA configurability
- Measuring performance numbers

Sample Results

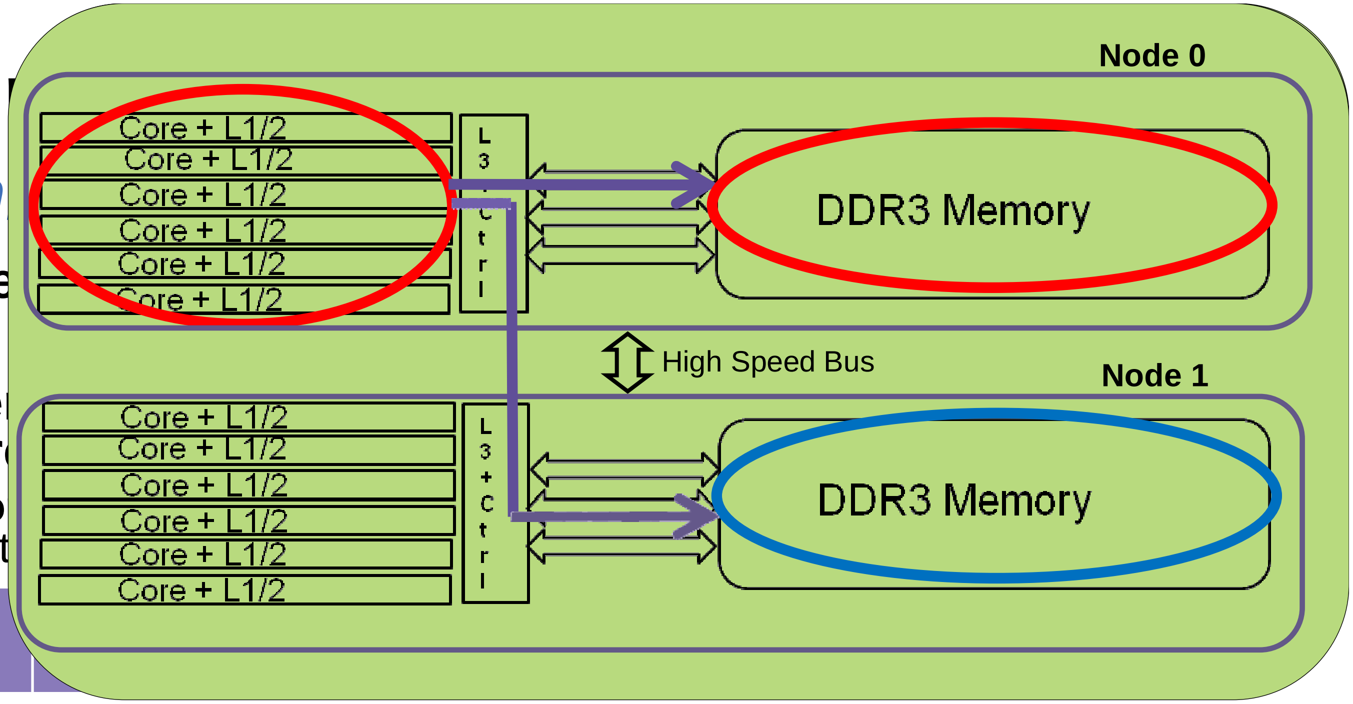
Minimizing variability (other factors)

Making Ap

numactl com

- **No Code change**

- **Applications we consistency in r**
  - **Use numactl co** shared memory t



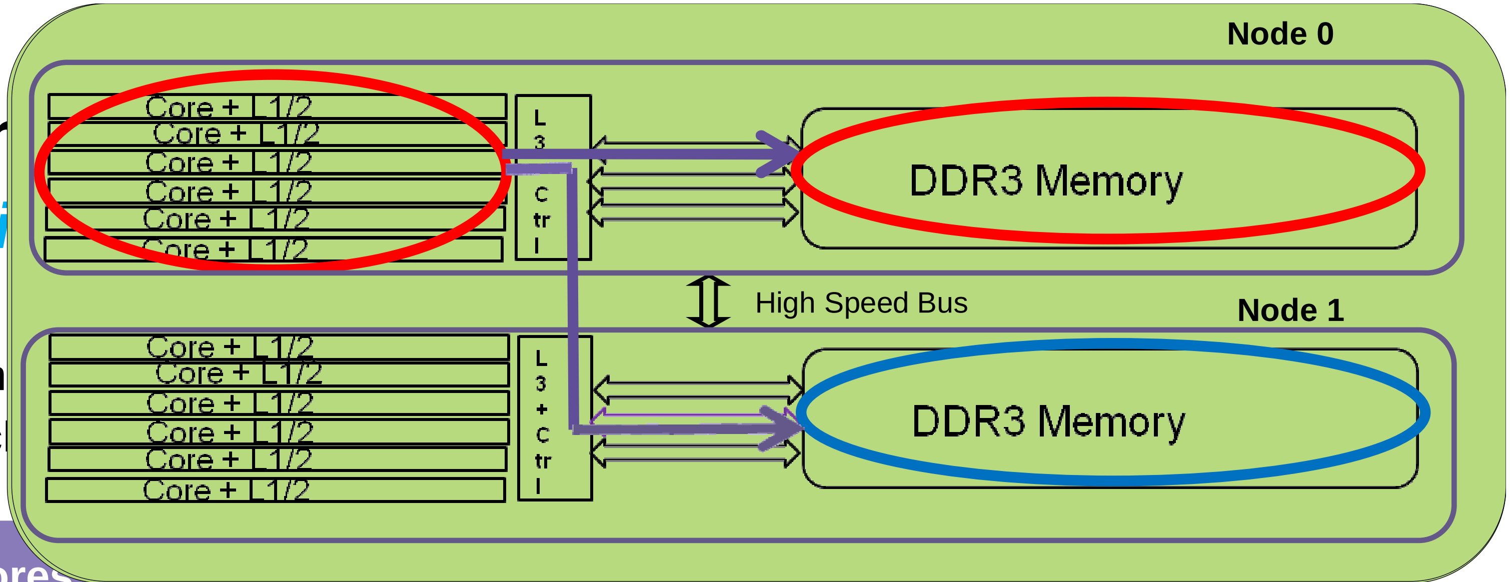
Numactl option	
-cpunodebind	Only execute process on the cores of the selected node
-membind	Only allocate memory from selected nodes. Jobs will start swapping when enough memory is not available on these nodes
-preferred	Preferably allocate memory on node. If memory cannot be allocated there, fall back to other nodes

- **LSF and UGE schedulers made NUMA aware**
  - In-house scheduler tool developed to achieve numactl functionality

Implemen

Cluster Setti

- Scheduler m
- SGE/LSF sc



Job Type	Cores Required	Memory Required	CPU-Bind	Memory-Bind
Single Thread	1	<= Total RAM on node	Y	Y
	1	> Total RAM on node	Y	Y (preferred)
	1	= Total RAM on machine	Y	Y (preferred)
Multi Thread	**N/A**	**> Total RAM on machine**	**Job remains Pending**	
	<= number of cores on node	<= Total RAM on node	Y	Y
	<= number of cores on a node	> Total RAM on node	Y	Y (preferred)
	> number of cores on node	**<= Total RAM on machine**	**Get entire machine**	

SYNOPSYS 25 years

Hardware/Software Settings
Recap

- Hardware BIOS settings available to reduce variation

- Tests are run using NUMA aware schedulers
  - No change was made in application code
  - Scheduler handles binding appropriately

- Pre-requisites for End Users
  - Resource requirement for tests be specified upfront
    - Memory requirement
    - Core requirement for Multi-Thread jobs

- **Better Utilization of resources**
  - Running up to 4 jobs per node
  - Faster throughput

Agenda

Problem Statement

Methodology

- NUMA hardware settings
- Cluster setup
- **Measuring performance numbers**
  - Creating Baselines
  - Regular Benchmark test runs
  - Re-establishing Baselines

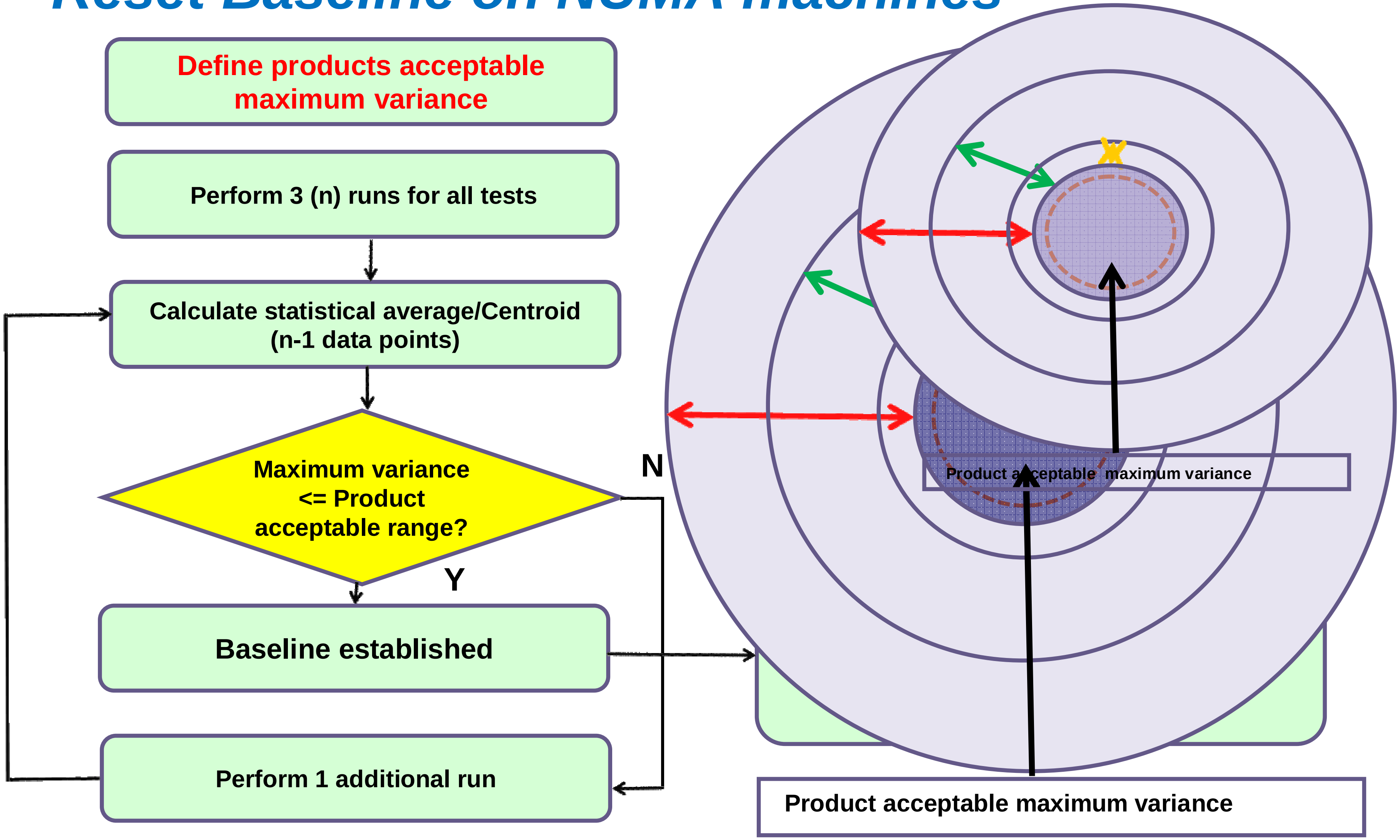
Sample Results

Minimizing variability (other factors)

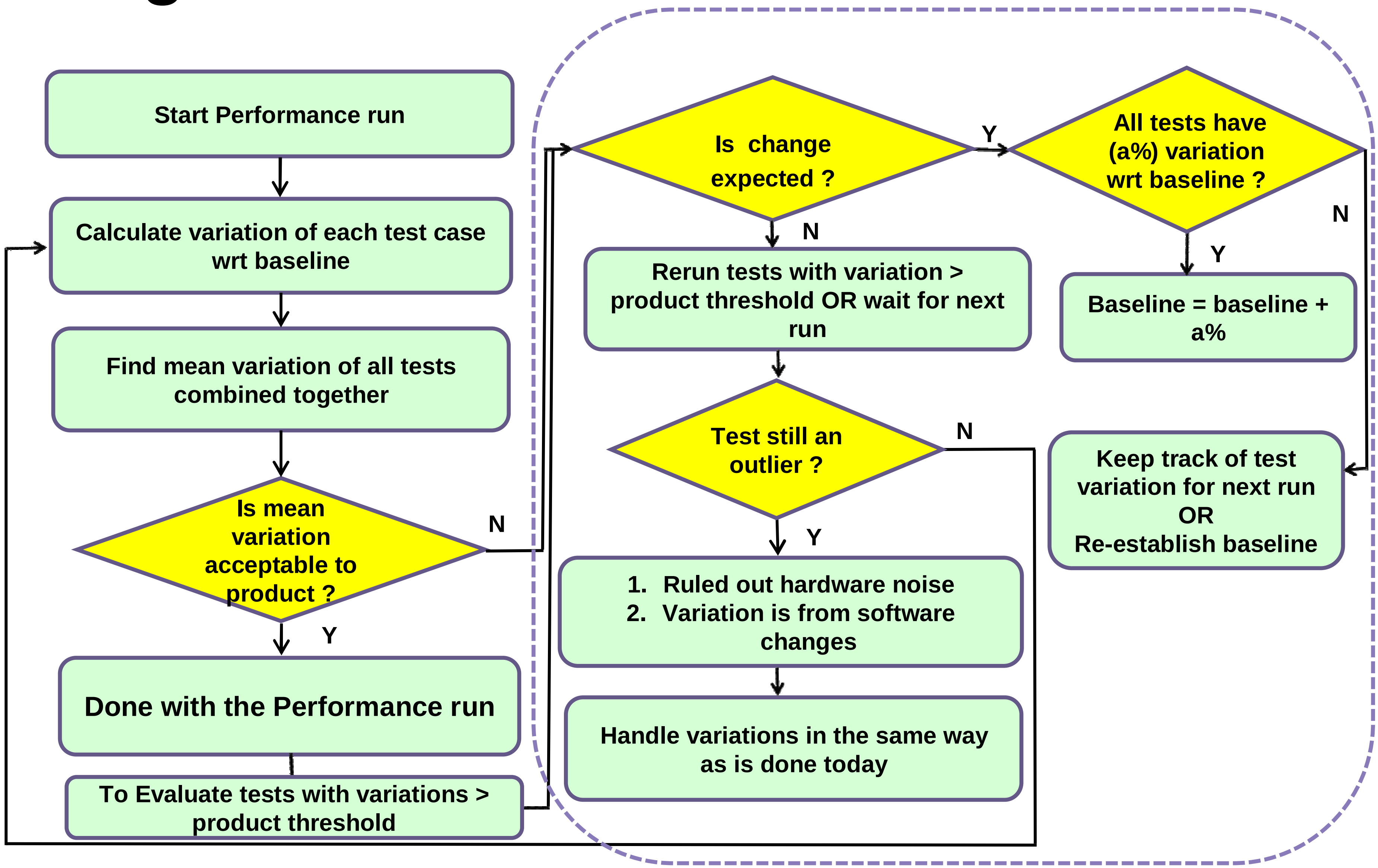
SYNOPSYS® 25 years

Methodology –Creating Baselines

Reset Baseline on NUMA machines



Regular Benchmark Runs



Re-establishing Baselines

- From release to release
- Large number of test cases have variations above the products acceptable threshold OR products curve has shifted
- If all tests show similar amount of variation with respect to the baseline

Agenda

Problem Statement

Methodology

Sample Results

- Single Thread tests
- Multi-thread tests

Minimizing variability (other factors)

Questions to Answer
Single-Thread/Multi-Thread tests

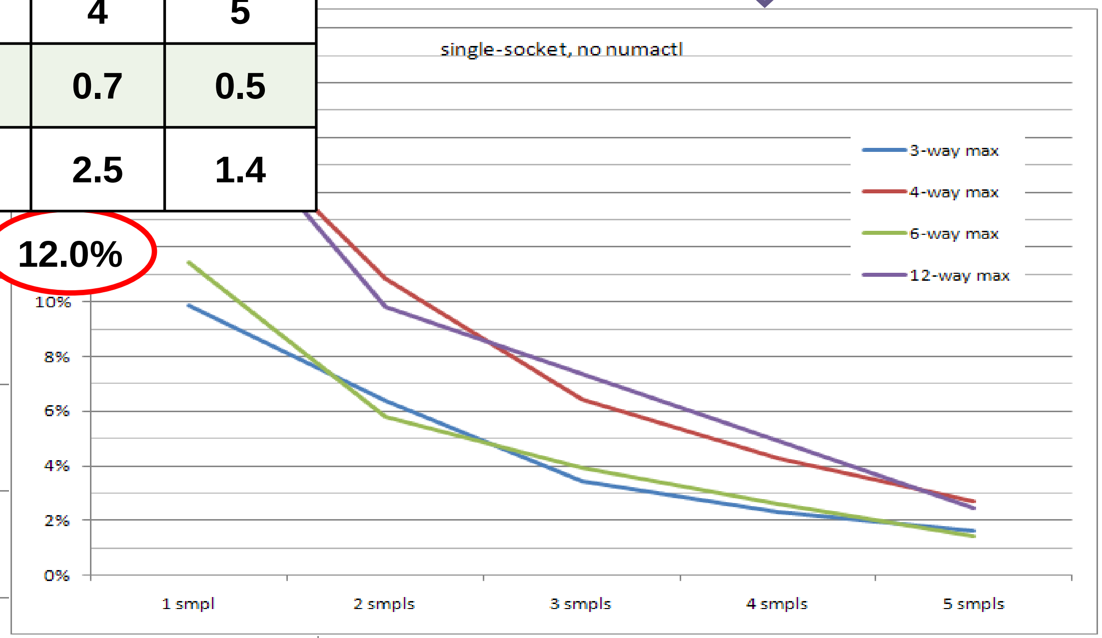
- What is needed to achieve "statistical stability" on NUMA machines?
- How to minimize number of runs to characterize and confirm performance?
  - Given a baseline, can single runs be useful to confirm performance?
  - Single runs need to be dominant mode for regular runs
- How many tests can be run per node?

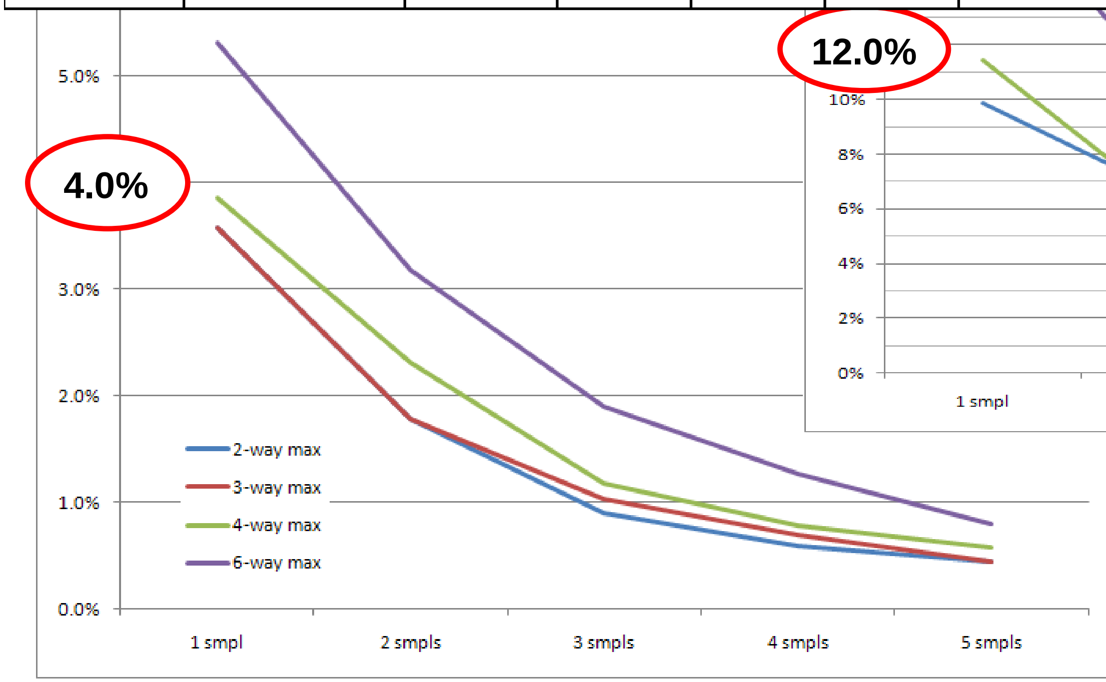
Baseline: Single-Thread tests

Without Binding - Variance is High, More sampling

Binding	Jobs per Node	Max Variance (%)				
		Number of Samples				
		1	2	3	4	5
Y	4	3.8	2.4	1.1	0.7	0.5
N	3	11.5	5.8	3.9	2.5	1.4

- No binding





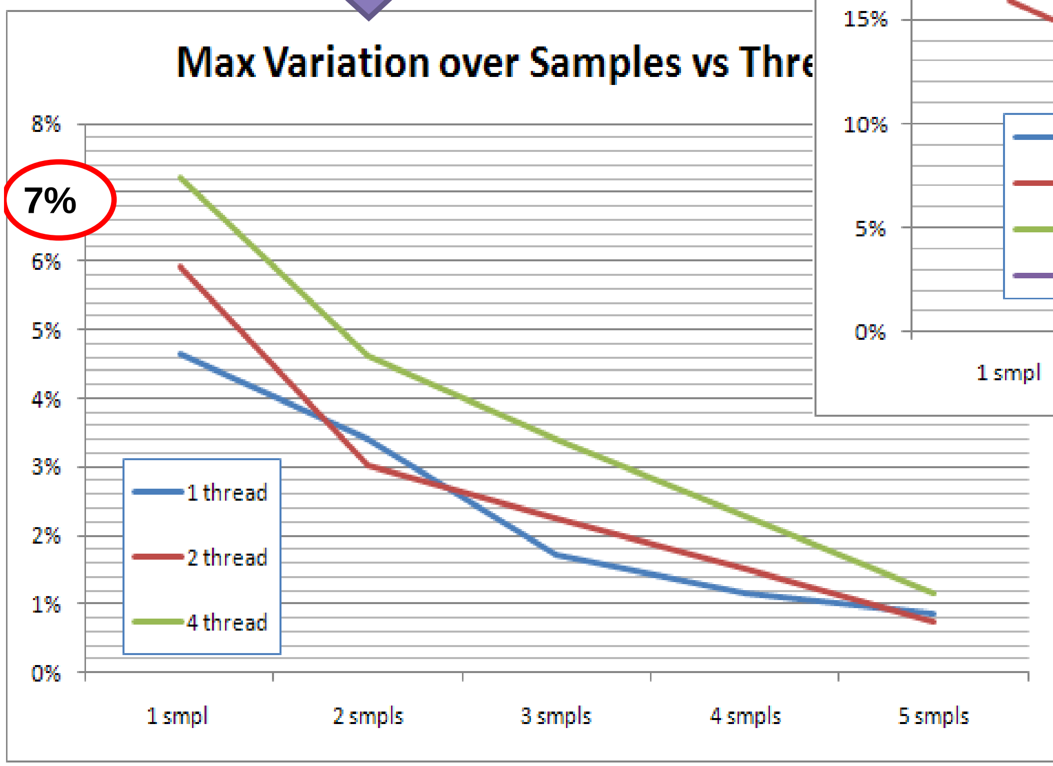
Without binding, maximum variance unacceptable unless taking 4+ sample runs using 3 cores

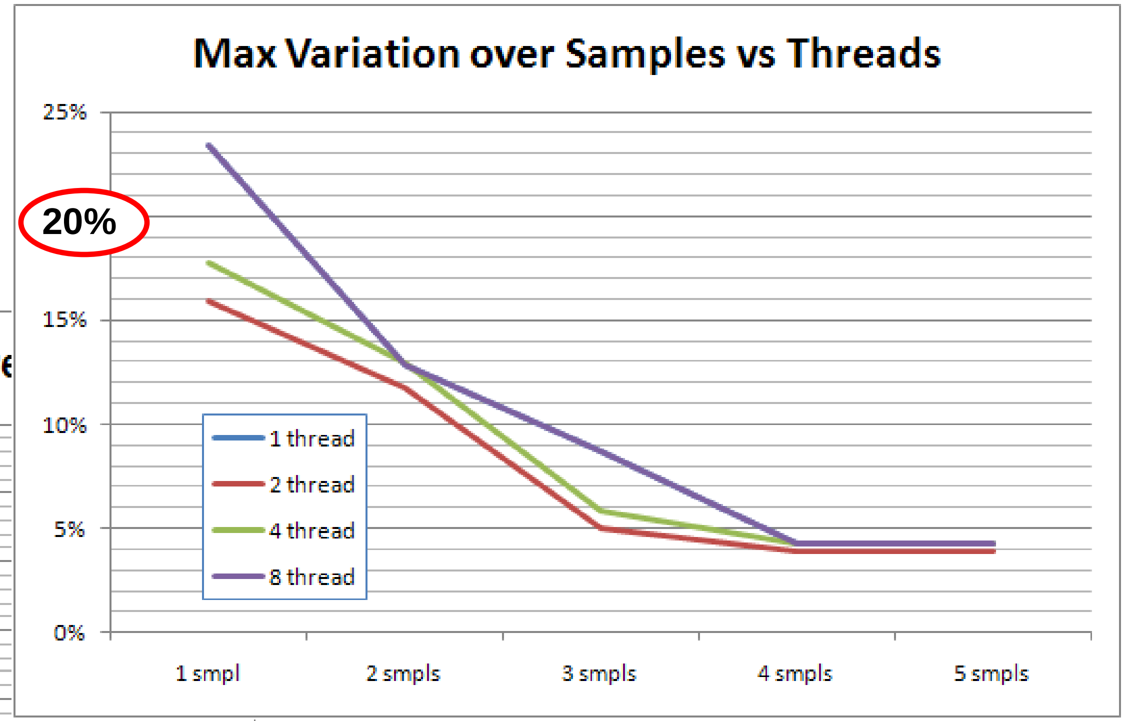
Small variation from 2 to 4 cores in use

Baseline: Multi-Thread tests

Without Binding, variance is High

- Core + Memory binding





- NO core or memory binding

Without binding, maximum variance unacceptable

Agenda

Problem Statement

Methodology

Sample Results

Minimizing variability (other factors)

Minimizing Variability

Other Factors

- Run tests on local disk to eliminate/reduce network I/O

- Vendor specific BIOS parameters for reduced variability.

  For example, for HP machines

BIOS	Setting for Benchmark tests
HP Power Regulator	Static High Performance
HP Power Profile	Maximum Performance
CPU Idle Power State	C6 state

- Linux “libnuma” library

Conclusions

- Making applications NUMA aware during run time shows great advantage in achieving reasonable variability with single benchmark runs

Contacts

- Vinnie Kasula vinnie@synopsys.com
- Sangeeta Aggrwal sangeet@synopsys.com

Q&A


Thank You
SYNOPSYS 25 years