System Management Interrupts and Their Effects

Manjeet Bhatia
Vishesh Jain
Konstantin Macarenco
Josef Mihalits
Stephanie Sohn

Introduction

- System Management Interrupt (SMI) :
  - Entry into System Management Mode (SMM)
    - More privileged than operating systems and hypervisors
    - Invisible as well
  - Causes all processor cores to stop and enter
- Reasons for SMIs/SMM:
  - Power throttling
  - Hardware emulation
  - RIMM (security)
  - System health checks

Known SMI Effects

- Device and OS interrupts preemption
- Severity increases with number of cores
- Jitter in timer interrupt handling (short SMIs)
- Missing of timer ticks (long SMIs)
- More time spent in higher C-states (more energy consumption)
- Device driver impacts (e.g audio and video distortions)
- Time scaling discrepancies (e.g. OpenSSL experiment)
- Performance degradation
- Loss of throughput

Experiments/Benchmarks

- Linux compilation and CPU bound
- Netperf (Networking)
- LMBench
- NBench (Integer and Floating Point Computation and Memory Access)
- Message Passing Interface (MPI)

Hardware

Types of nodes:

- Seven 4 core nodes (all other smitest machines)
- One 8 core node (smitest3)

Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, cache size: 8192 KB, 12Gb RAM

Gigabit ethernet devices

SMI Length Investigation

Goal :

Confirm built-in SMI driver’s latency checker

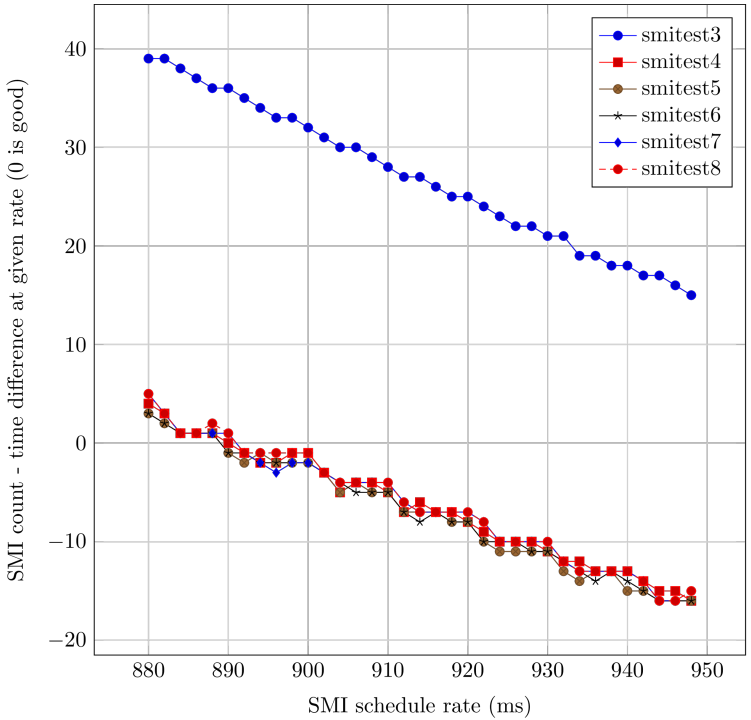
Approach :

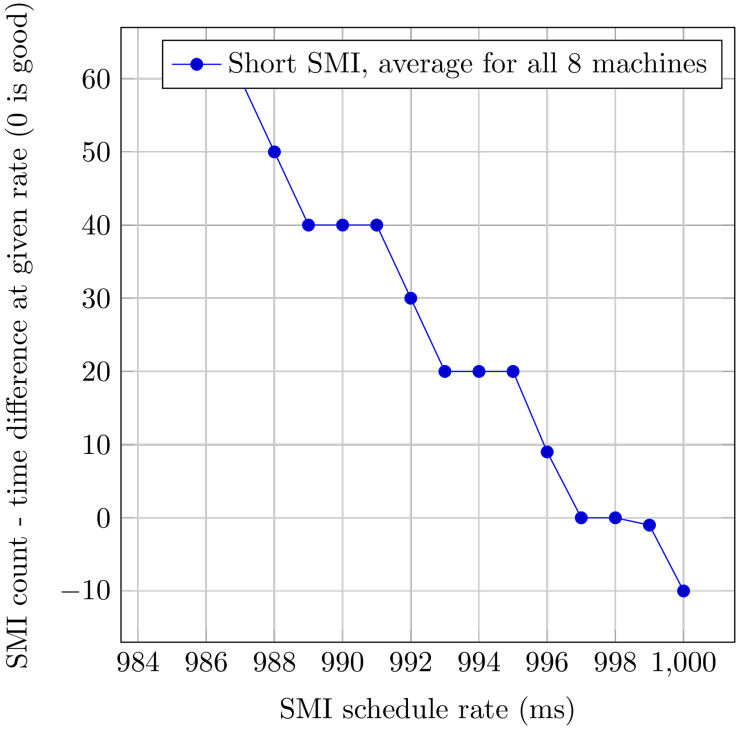
- If SMIs length + schedule_interval = 1 sec (exactly) i.e. one SMI per second, then after any N seconds, difference time (seconds)
- N - smi_count = 0

Methodology :

Test smi_count - time for range of schedule intervals

Long (left graph) and Short (right graph) SMI Length Deviation





SMI rate: SMIs are scheduled every x ms

Deviation: Difference between SMIs occurred and time elapsed in seconds

SMI Length Conclusion

All eight nodes have the same length short SMI which is:

- Short SMI 1-3 ms approx - result is more precise than with built in driver latency checker (5ms)

Nodes 4 - 8 SMI length

- Long SMI 100-110 ms approx - built in driver 100 - 110 ms

Deviation:

Smitest3 long SMI length according to the test: 45 - 50 ms.

Built in driver 100 - 110 ms.

System Time Delay

Reason: during SMI length testing, side effect was noticed - all 8 nodes have different wall time:

- June 6 18:46:50 smitest
- June 6 19:45:09 smitest2
- June 6 18:54:42 smitest3
- June 6 16:26:14 smitest4
- June 6 17:43:26 smitest5
- June 6 18:37:45 smitest6
- June 6 13:12:27 smitest7
- June 6 13:00:57 smitest8
- June 6 18:44:44 wyeasthead - gateway (real time).

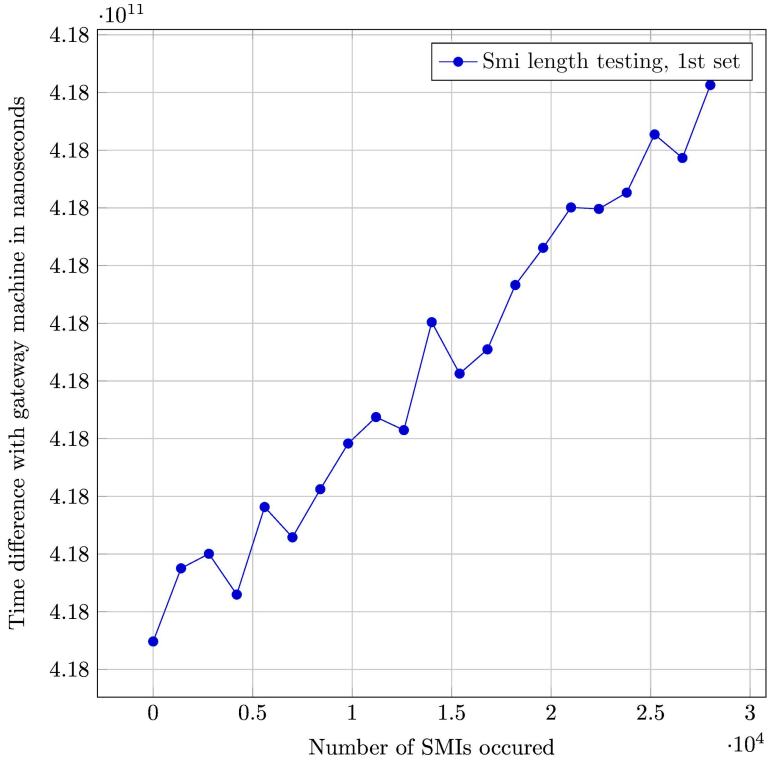
All nodes except wyeasthead:

- no internet connection
- therefore no time synchronization

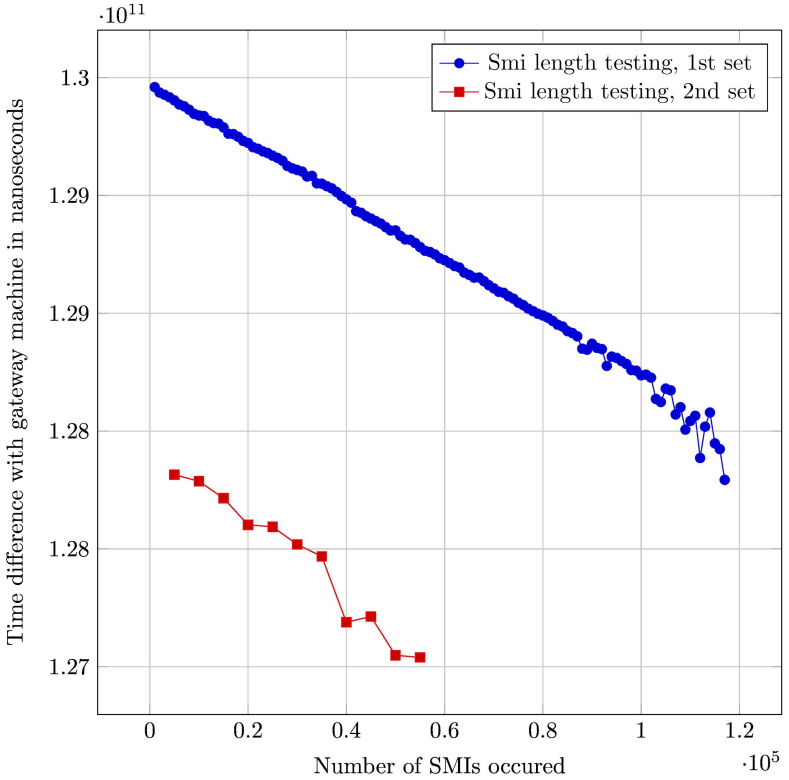
Methodology:

To test if wall time is affected by recurring SMIs, time the difference between test machine and gateway every 1000 SMIs.

System Time Delay



Difference between smitest6 and gateway (ahead)



Difference between smitest and gateway (behind)

System Time Delay Conclusion

- System time is affected by recurring SMIs, it freezes for some fraction of SMI length which is not noticeable for short periods of time, yet has accumulative effect
- This can potentially lead to synchronization problems

CPU and Linux Kernel Compilation

- Nbench (set of CPU bound tests) benchmark to test CPU bound performance
- Disadvantage: Nbench created in 1996 - not suited for multi-core performance testing, but it was chosen because it doesn’t require any external dependencies and no installation (system access limitation e.g. no root access),
- Linux compilation well known way to test overall system performance; can be compiled with make -j n option (n is number of available cores)
- Average of 20 runs for each test

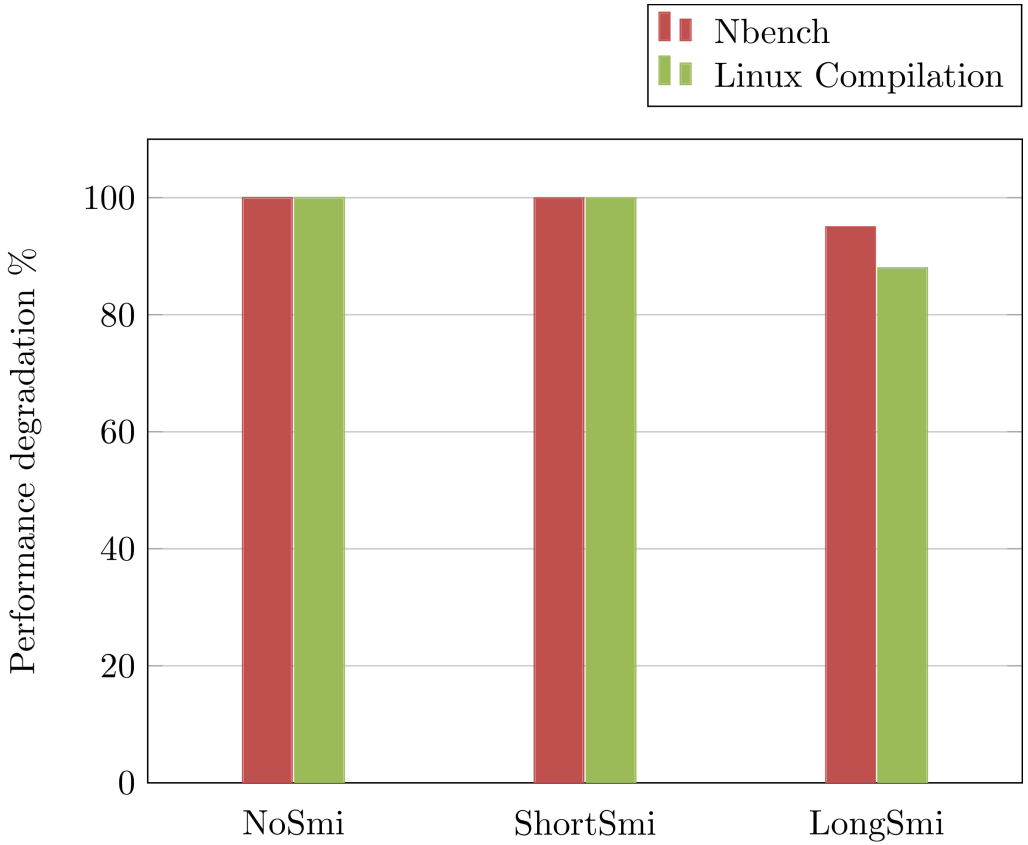
Methodology:

Throughput degradation is measured

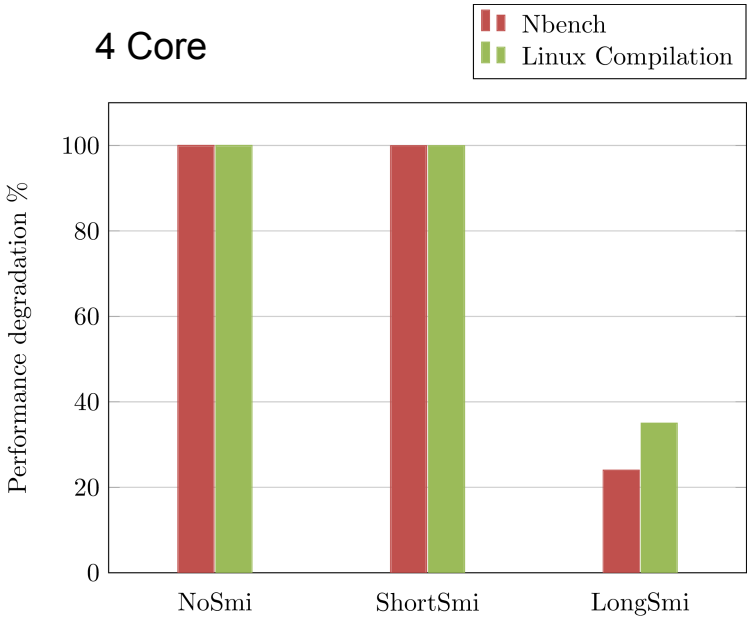
CPU and Linux Single Core

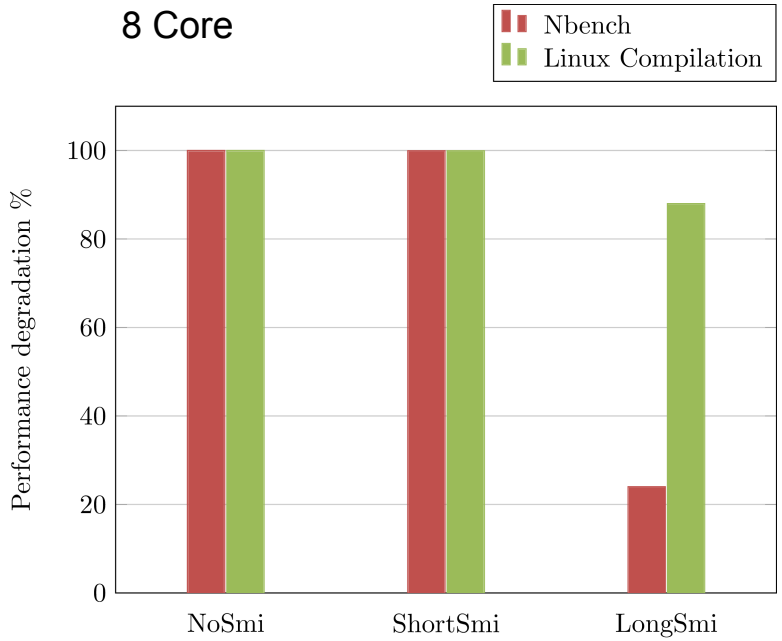
Test are unaffected by short SMIs
1 SMI per second (at most 0.05 performance degradation)

Long SMI have more significant effect:
Nbench 2% degradation.
Linux compilation 12% degradation.



CPU and Linux Multicore (4/8 cores)





Throughput degradation: 76% Nbench, 64% Linux compilation; long SMI (100ms/1s)
0.05% At most; short SMI (2ms/1s).

CPU Bound Conclusion

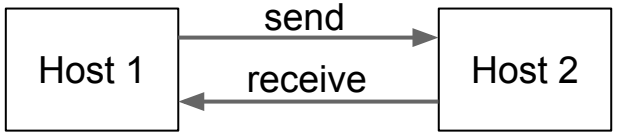
- Long SMIs show worse performance degradation for multi-core vs. single core (especially CPU-bound)
- Short SMIs have very little effect on performance. Tests sometimes run faster with short SMIs enabled.
- Re-confirmed Delgado/Karavanic results

Netperf: Overview

Goals:

- Investigate effects of SMIs on network throughput
- TCP and UDP with long vs short SMIs

Methodology:



- Each test ran for *1 min*
- Average of *5 runs*

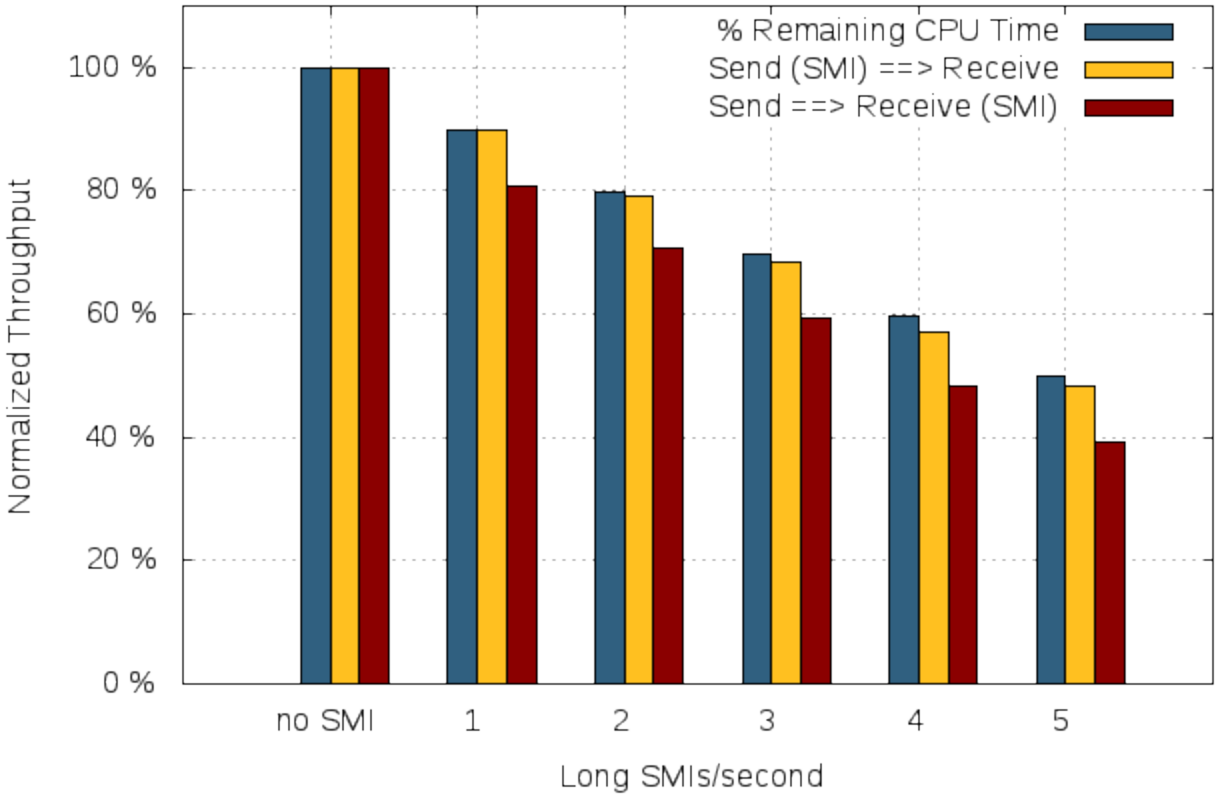
Netperf: Data Transfer, Long SMI

# SMIs / Second	GB Sent via TCP Send (SMI) ➡ Recv	# Retransmissions	GB Sent via TCP Send ➡ Recv (SMI)	# Retransmissions
0	5.60	0	5.60	0
1	5.05	0	4.53	49865
2	4.43	0	3.95	45418
3	3.82	3	3.32	37518
4	3.21	2	2.71	28775

# SMIs / Second	GB Sent via UDP Send (SMI) ➡ Recv	GB Sent via UDP Send ➡ Recv (SMI)
0	5.71 ⇒ 5.71	5.71 ⇒ 5.71
1	5.13 ⇒ 5.13	5.71 ⇒ 5.09
2	4.46 ⇒ 4.46	5.71 ⇒ 4.48
3	3.84 ⇒ 3.84	5.71 ⇒ 3.86
4	3.22 ⇒ 3.22	5.71 ⇒ 3.19

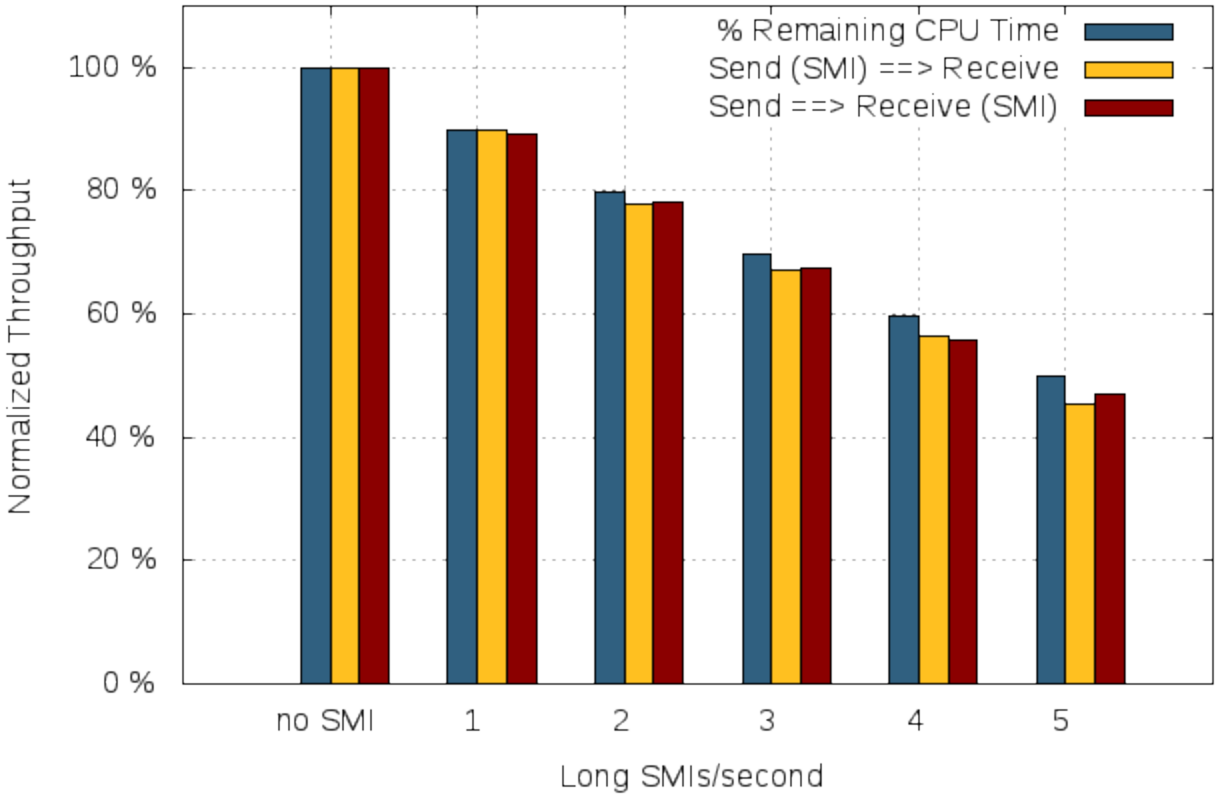
Std. Deviation < 102 MB

Netperf: TCP Throughput, Long SMI



Std. Deviation < 1.5 %

Netperf: UDP Throughput, Long SMI



Std. Deviation < 1.2 %

Netperf: Throughput, Short SMI

- TCP, UDP
- 1, 10, 100, and 500 SMIs / second

⇒ No noticeable performance degradation.

Netperf: Conclusion

- No stability issues
- Short SMIs have no noticeable effect

- Sending TCP, UDP: Throughput degradations match SMI latency
- Receiving TCP: Additional degradation due to mechanics of TCP/IP protocol
- Receiving UDP: Increased risk of packet loss

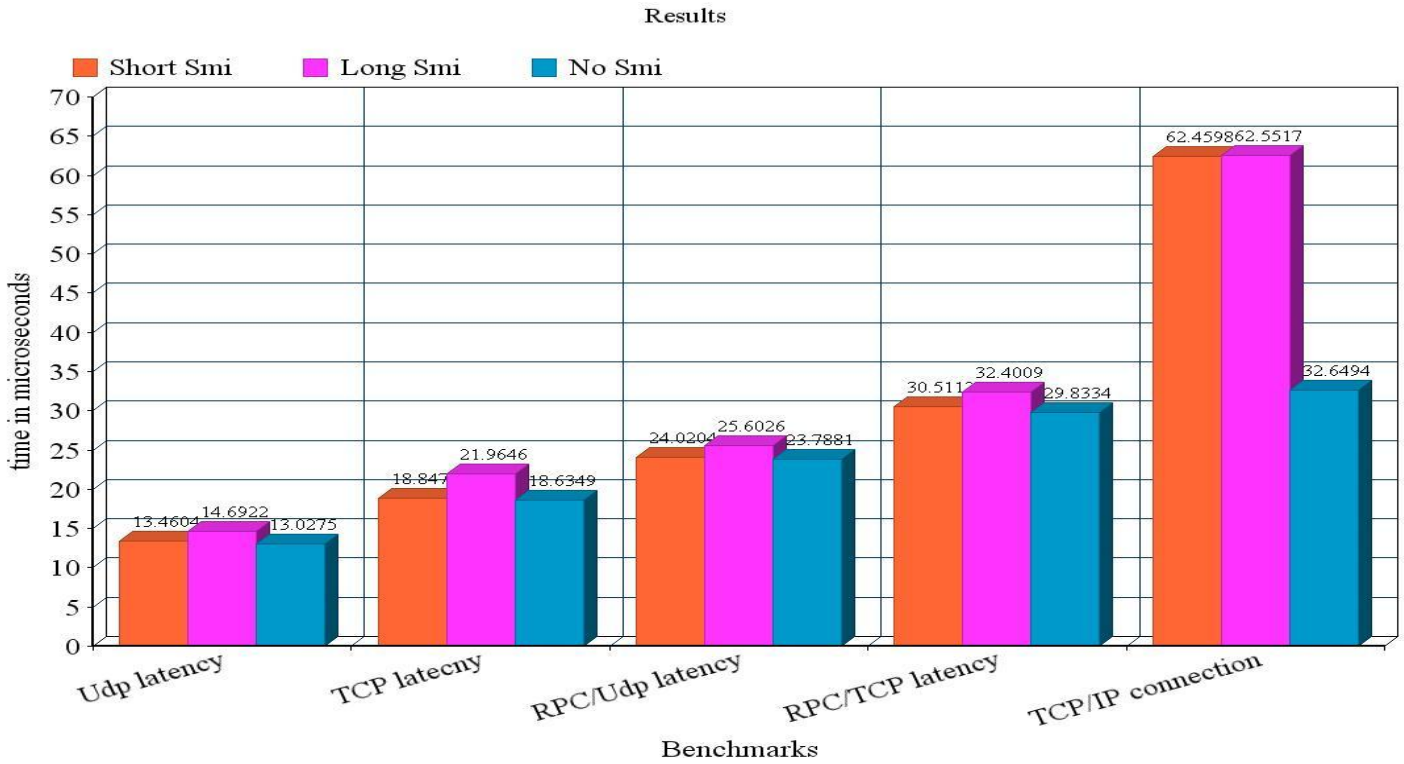
Results confirm the findings reported by Delgado and Karavanic.

LMBench

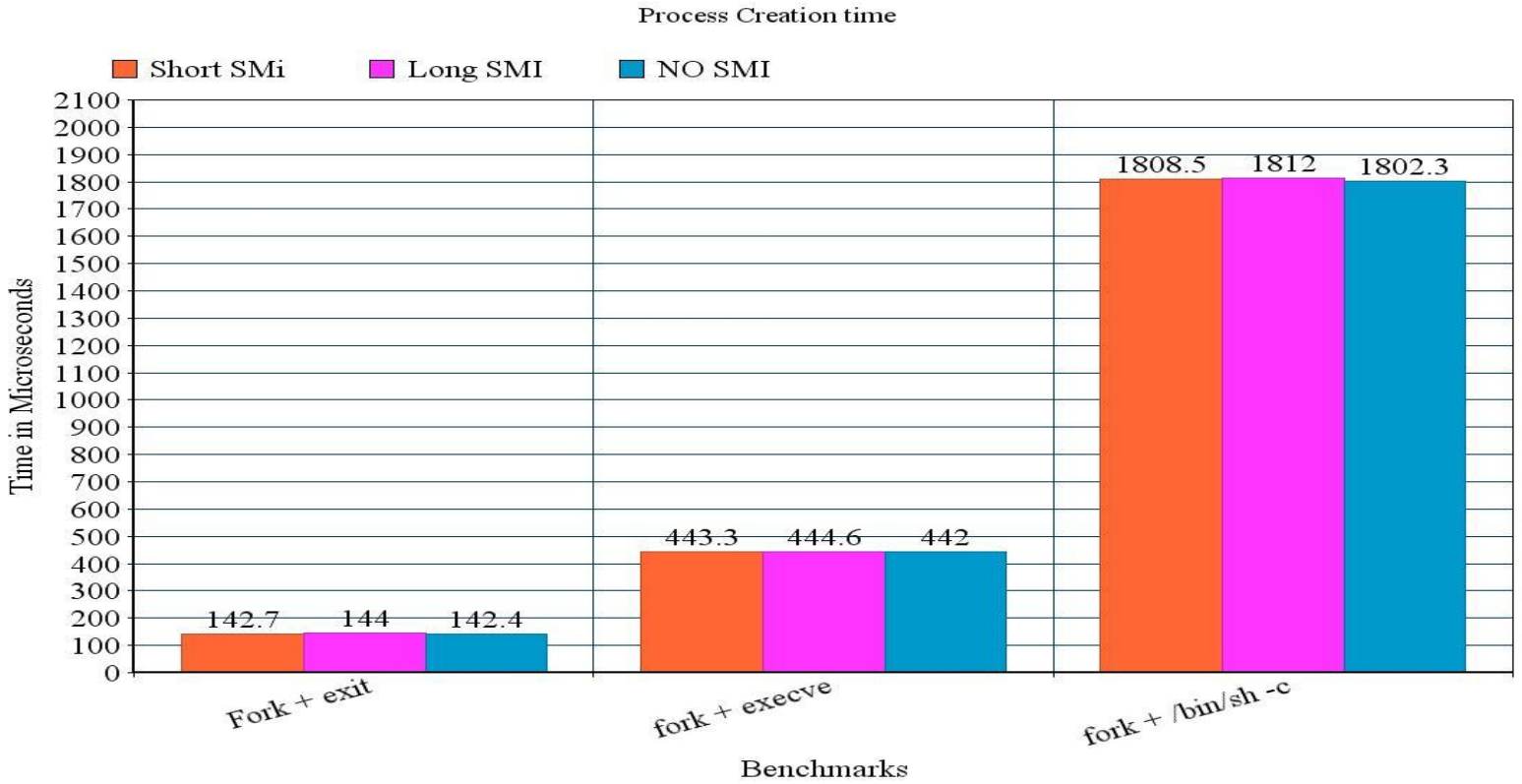
Goals

- To measure connection establishment time between processes to initiate communication locally
- To measure the effects of small and long SMIs on that time by running lmbench which will tell the time in microseconds
- To measure latency of processes with SMI effects
- To measure effects of SMI on Pipe Bandwidth

Connection Latency



Process Latency



Pipe Bandwidth

Pipe Bandwidth is the amount of data transfer when result of one command is passed as input to other command and the transfer rate of data will affect the performance of that operation.

Short SMI	Long SMI	No SMI
1970 MB/sec	1750 MB/sec	1983 MB/sec

LMBench Conclusions

- In process communication Short SMIs does not affect the performance it is almost same as no SMIs
- But in TCP/IP connection there is huge difference between short SMI and no SMI
- Pipe bandwidth is also affected by SMI for long SMI it has the difference of almost 110 Mb/sec when done with no SMI
- For process creation and exit for short SMI is same as no SMI had 2 microsecond difference with long SMI

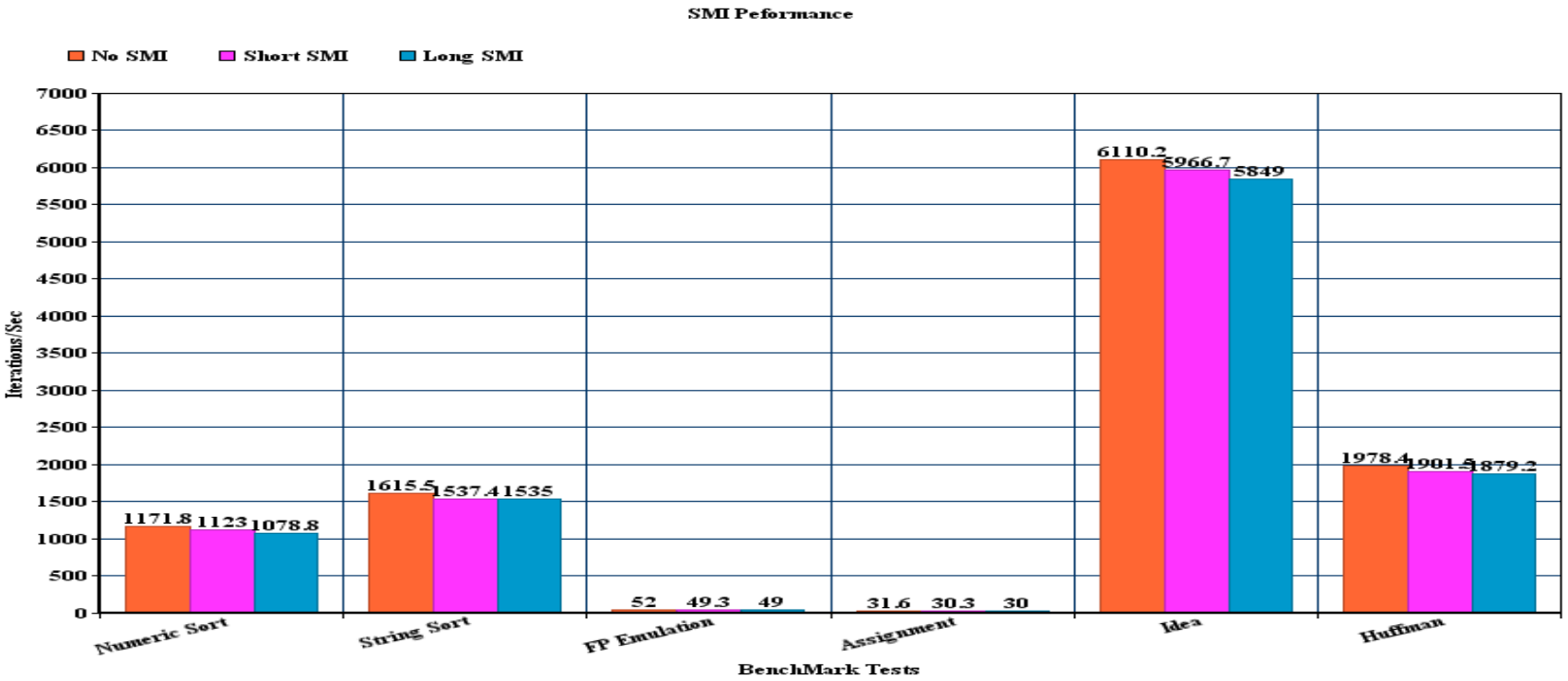
NBench Overview

- C programs to measure integer, floating point and memory performance of the system

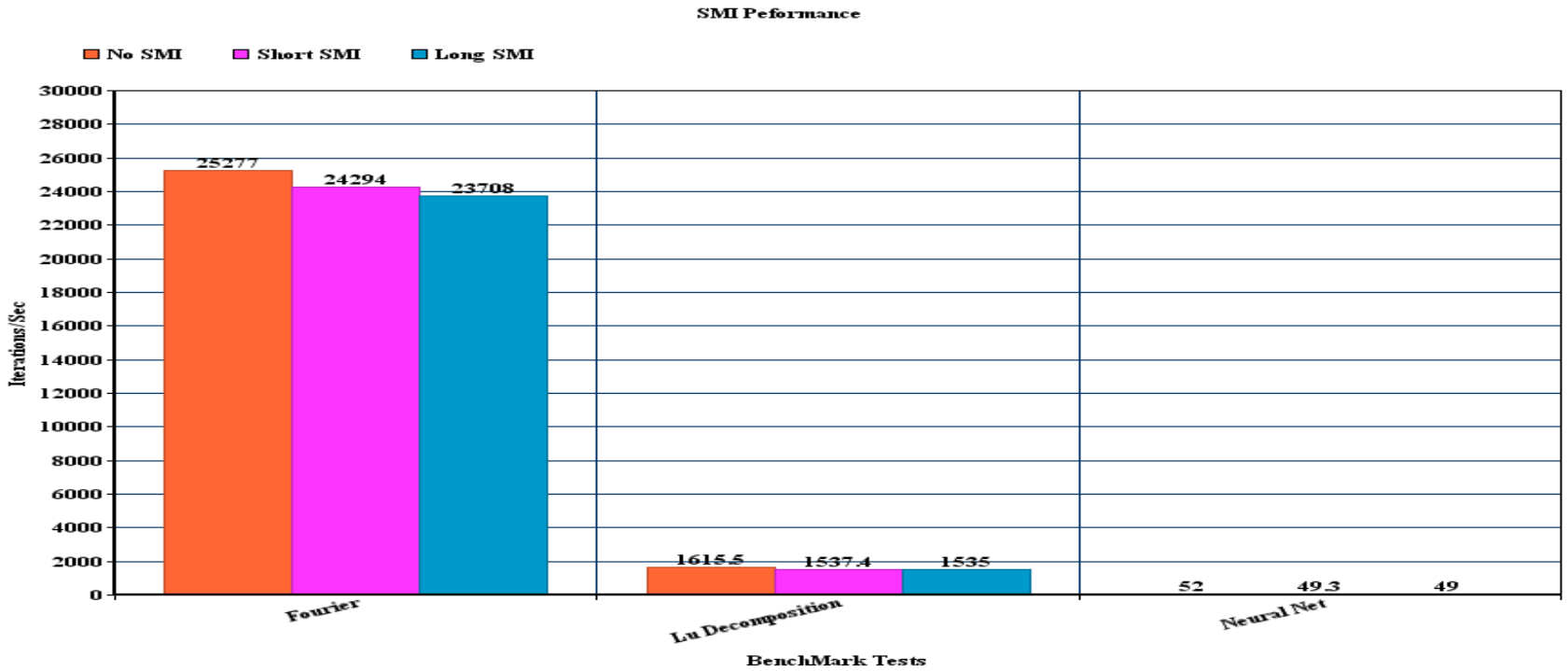
- Important for testing memory access and computation work

- Run each program for a time GLOBALMINTICKS
  - After program has run for this minimum time, next program starts

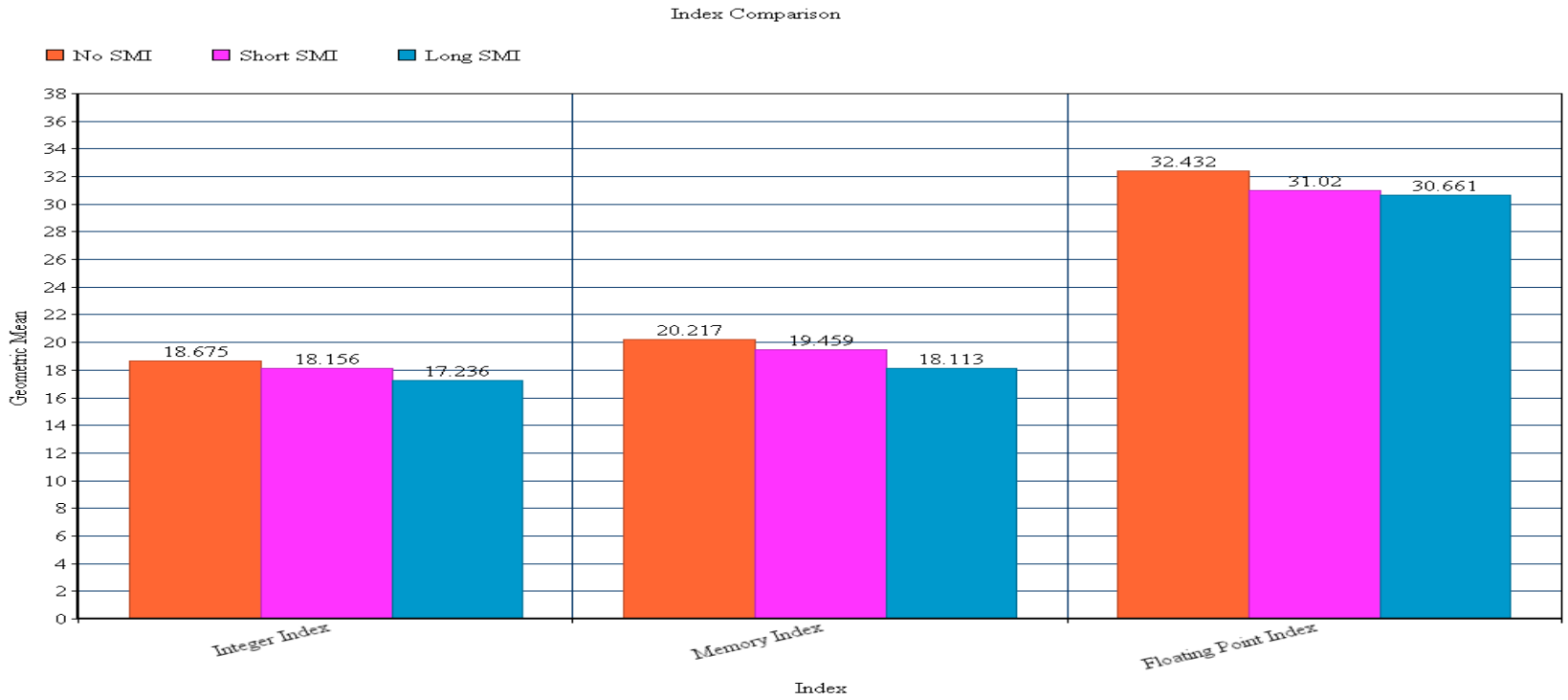
NBench Integer And Memory Computations



NBench Floating Point Computation



NBench Index Comparison



NBench- Performance Impact

- Integer Index Calculation Performance Reduction

  Short SMM - 2.7%

  Long SMM - 7.7%
- Memory Access Index Calculation Performance Reduction

  Short SMM - 3.7%

  Long SMM - 10.4% (Maximum Degradation)
- Floating Point Calculation Performance Reduction

  Short SMM - 4.3%

  Long SMM - 5.4%

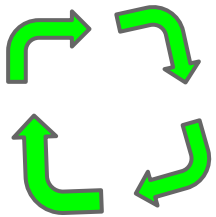
NBench Conclusions

- Long SMI consume CPU cycles resulting in interruption of the application
- Only average computation of the application is affected when scheduling regular short SMI
- Some individual tests ran better than with no SMIs enabled - no significant interruption of the program
- Integer, Memory and Floating Index reduced - application computation is affected
- Hence SMM pauses the execution of the host software

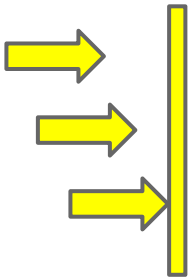
Message Passing Interface (MPI)

- Ring, Barrier, AllReduce, AlltoAll
- Varying number of iterations (1K vs. 1M)
- 2,4,8 machines communicating or
- 1 machine with 2,4,8 processes communicating
- SMIs settings - off, short (1 SMI per second, ~0.05 ms), long (1 SMI per second, 100-110 ms)

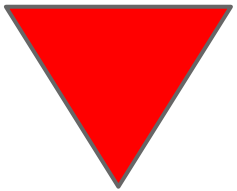
Ring



Barrier

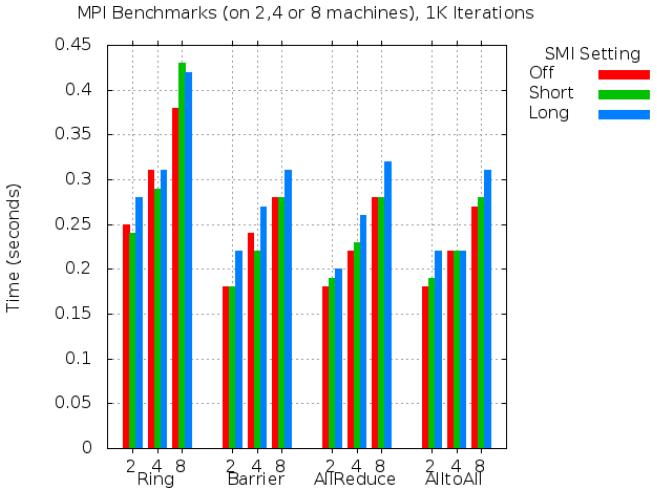


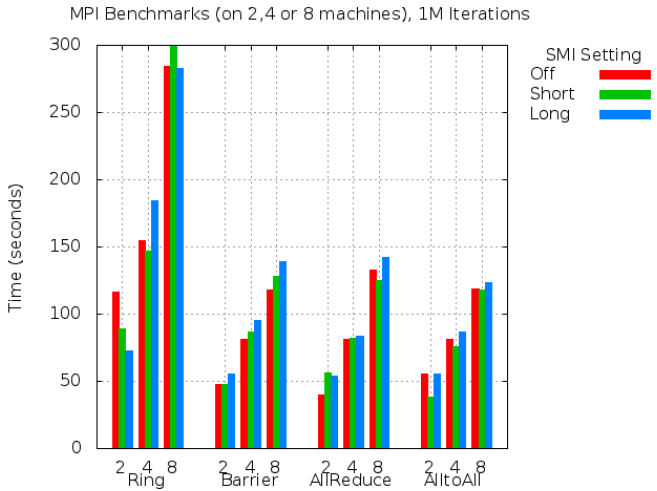
AllReduce



AlltoAll

Comparing Iterations

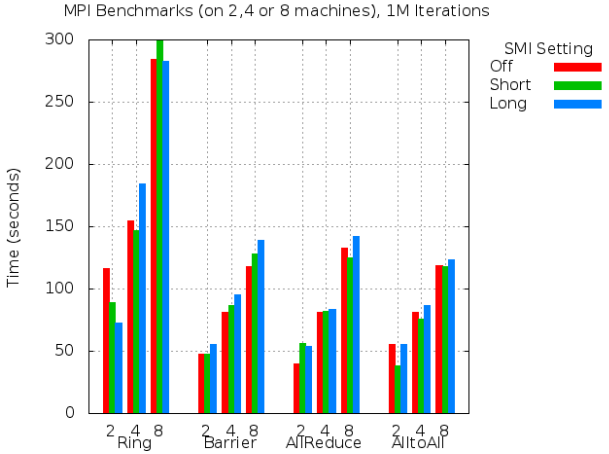




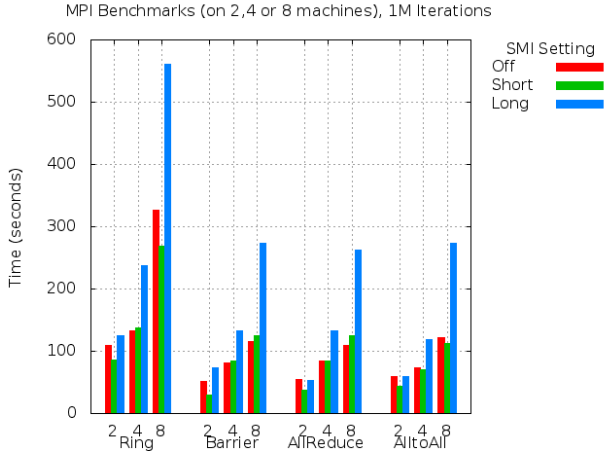
- SMIs enabled on 1 node; communication between 2, 4, or 8 nodes
- Randomness (runtimes can vary as much as 50% difference in runtimes)
- Not significant normalized differences between iteration amounts
- General trend - long SMIs increase time (~5-10%), short SMIs not always

Number of SMI-enabled Machines

SMIs on 1 machine

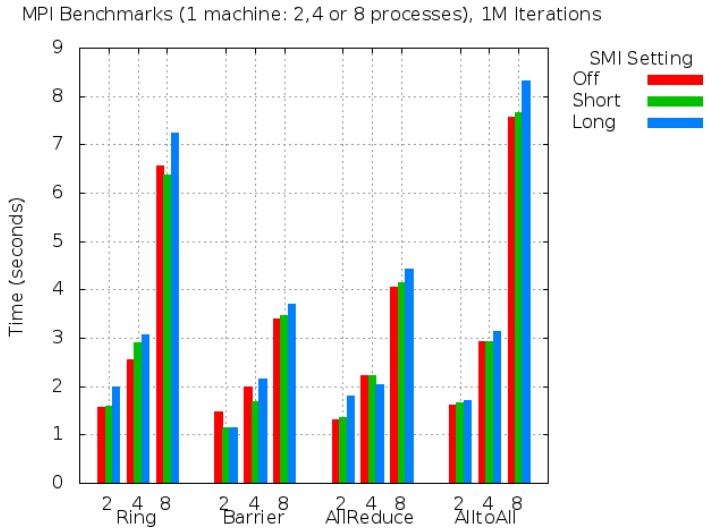


SMIs on all machines



- Significant differences between enabling long SMIs - on 1 machine vs. enabling upon all machines in use
- Not much for short - sometimes even shorter runtimes
- Overall 96% vs. 66% (100% is with SMIs off) - 1 machine vs. all machines enabled
- More drastic effect with more nodes involved for all machines enabled (2-90%, 4-61%, 8-47%)
- Barrier tends to be greatest affected, but similar amounts for all benchmarks (60-70%)

MPI on one machine, 2-8 processes



Benchmarks above ran on smitest 3 (8 core)

All to All Benchmark		smitest 3 8 core		smitest 5 4 core	
SMI Type	# Processes	% Remaining CPU clocks	% Normalized Time (100% for no SMIs enabled)	% Remaining CPU clocks	% Normalized Time (100% for no SMIs enabled)
Short	2	100.0	99.6	100.0	94.5
	4	100.0	100.3	100.0	101.3
	8	100.0	100.3	100.0	100.0
Long	2	93.3	97.2	100.0	97.1
	4	89.6	93.0	92.2	95.6
	8	90.6	92.6	90.5	75.7

MPI Conclusions

- Randomness affecting run times (more testing needed)
- Duration of run not necessarily important (<1 second vs. 100s of seconds)
- More machines with SMIs -> drastic additive effect
- Possibly smaller number of cores greater affected by long SMIs when number of processes > number of cores
- More testing in coordinating SMIs

General Conclusions

- Short SMIs seem not to matter
- Long SMIs give ~10% degradation on one machine (scales to latency of SMI)
- CPU bound degraded
- Application Computation Affected
- Networking degraded
- Multi-machine (MPI) - additive negative effects
- Multi-use and multi-machine - additive negative effects possible (combining CPU, networking, etc.)