In collaboration with Massachusetts Institute of Technology

Masdar INSTITUTE

Virtual Memory and Cache Driven Performance Evaluation of Workload in Multicore Processors

Rupesh Raj Karn
Masdar Institute of Science and Technology
Abu Dhabi, UAE

MIC615 Project

Date: 11th May 2017

OUTLINE

- Introduction
- Objective
- Experimental Setup
- Memory Limit
- CPU Resource Limit
- Conclusion

Introduction

- Multicore processors have been commonly used in industry and academia. They use virtual memory system for complex workload execution.

- A virtual memory system combines the physical memory and secondary storage device. It behaves as the computer system has more physical memory than is actually installed.

- Page, buffer, swap, active list, inactive list etc. are well known in virtual memory.

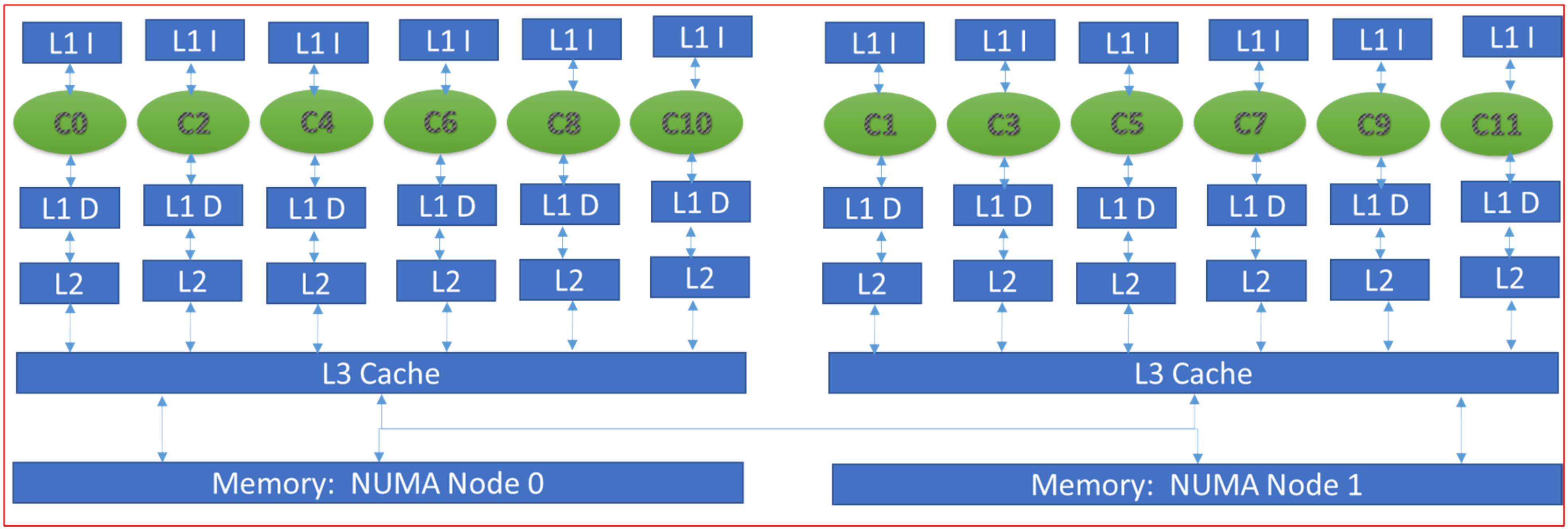
- Workload execution under limited memory causes performance degradation. Dynamic and intelligent memory provisioning based on processor's performance metric is required.

Objective

- Reduce the number of page faults and cache miss caused by unbalanced memory allocations.
- Measure the completion time and threshold memory size for workload execution.
- CPU power and frequency relation with memory metrics.
- Differences in virtual memory utilization and performance comparisons between compute intensive and memory intensive workloads.
- Learn the Linux tools that interact with hardware e.g page cache allocation per process, script to flush memory, cache operation modes as write back or write through, page table extraction etc.

Experiment Bench

- Tools used: Red Hat Linux, MATLAB , Minitab, Intel PCM, RAPL for processing.



Experiment Bench

- L1 and L2 caches are write-through and 8-way set associative. L3 cache is write back (associative unknown).

- Linux Command:

  dmidecode -t cache : Shows cache details

  cat /proc/cpuinfo : Shows processors details

  lscpu : Shows memory details.

Attribute	Specification
Processor	$2X$ Intel Xenon $E5-2630$
Cores Number	6 per socket. Total 2 socket.
Available Frequency(GHz)	1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3
L1 I-cache	32 $KB/core$
L1 D-cache	32 $KB/core$
L2 Cache	256 $KB/core$
L3 Cache	15 MB
Main Memory	16 GB $DDR3-1333$
Bus Speed	1333 MHz

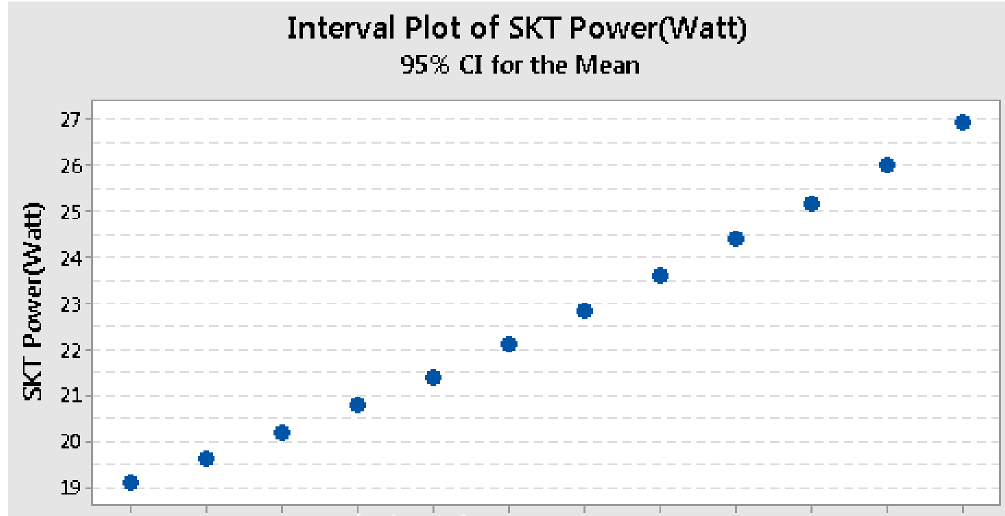
Processor and Memory Power

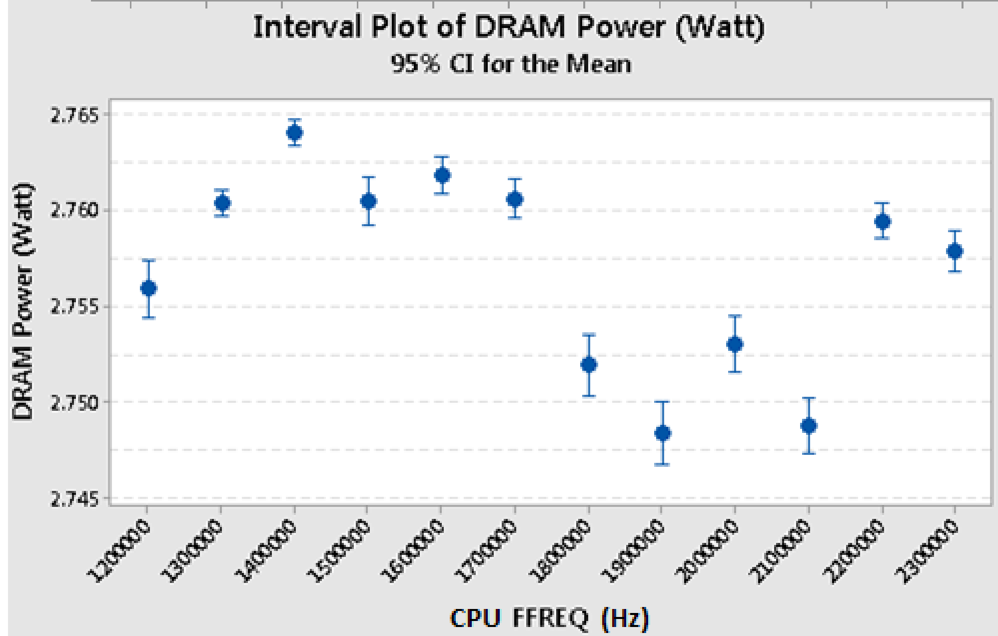
- Workload swaption from PARSEC benchmark is run and CPU frequency is switched many times between 1.2 GHz to 2.3 GHz.

- Memory clock frequency always stays at 1.6 GHz irrespective of CPU frequency.

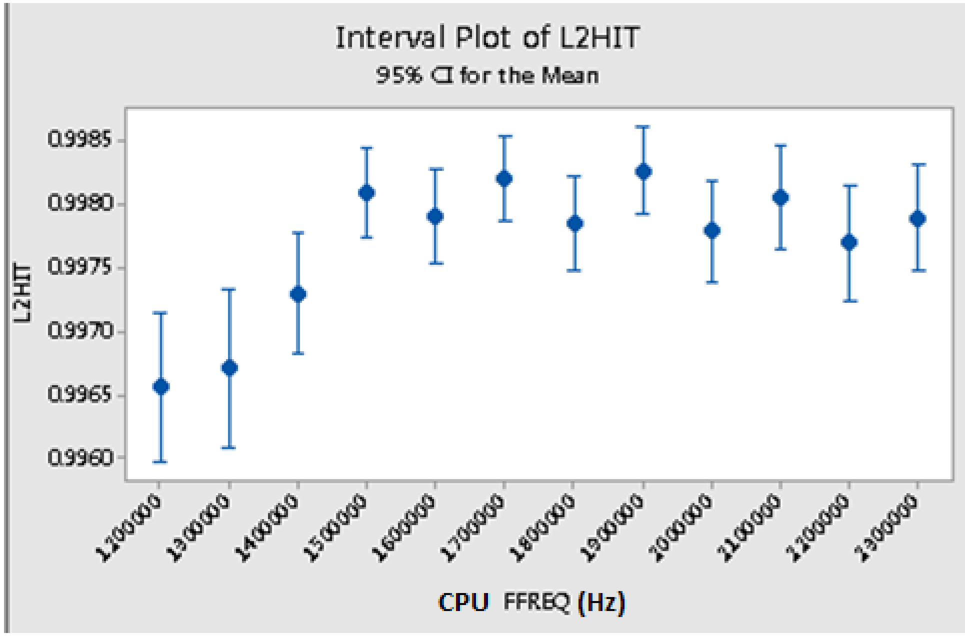
  Linux command: dmidecode --type 17

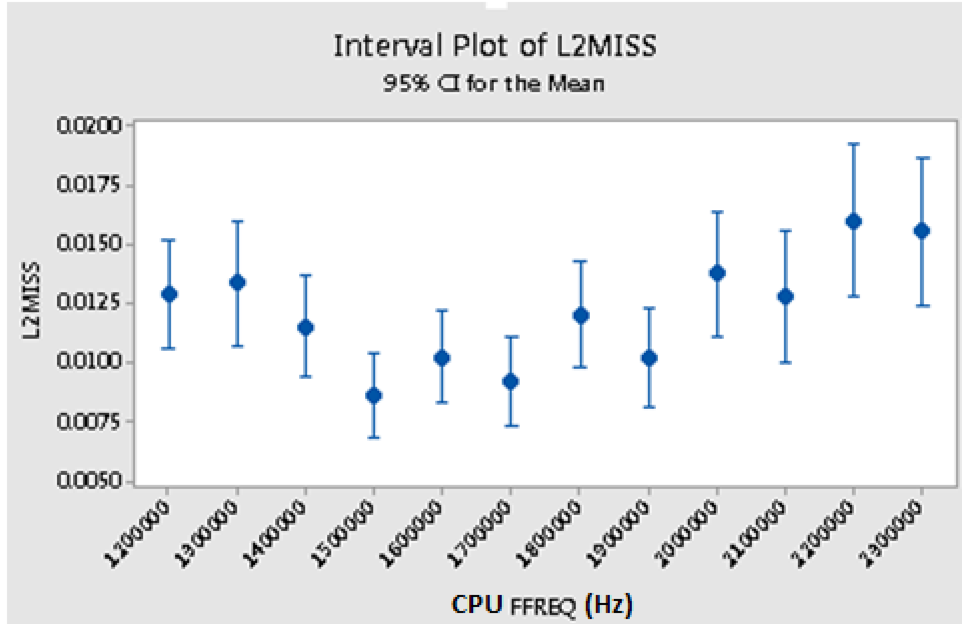
- No significant relation between CPU frequency and memory power consumption.



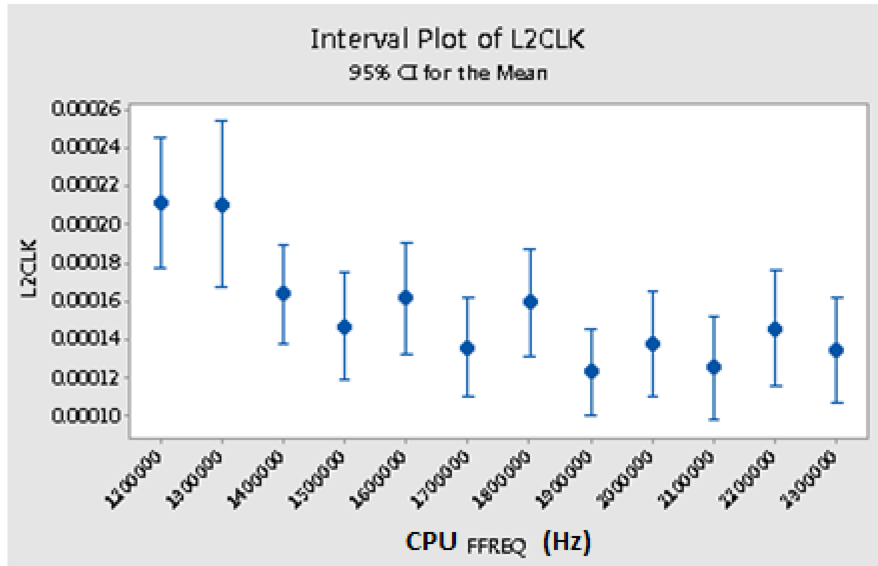


L2 Cache Metrics vs CPU Frequency

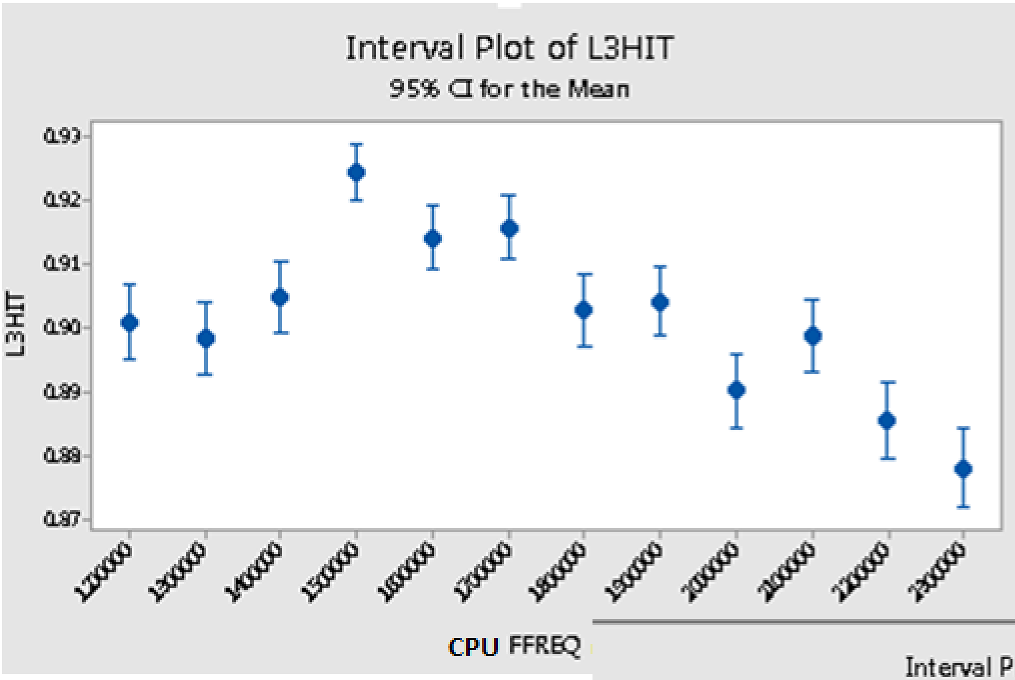


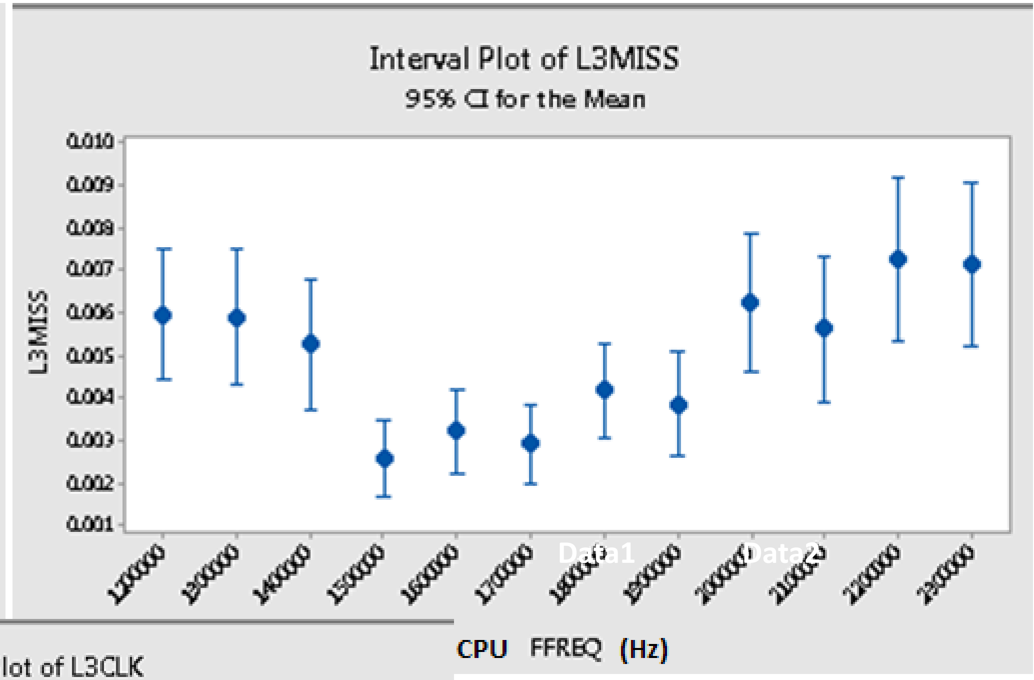


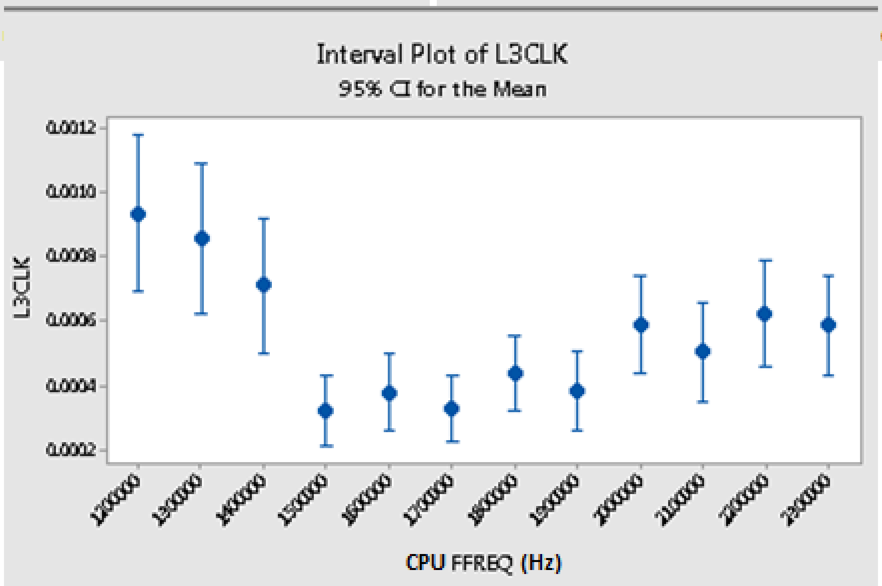
L2CLK : ratio of cpu cycles lost due to missing L2 cache but hitting L3 cache (like 10% time of 1 second is lost in searching the data at L2 ache but finding in L3 cache, so L2CLK = 0.1 because 10% of clock cycles are lost in searching for data in cache).



L3 Cache Metrics vs CPU Frequency







L3CLK : ratio of cpu cycles lost due to L3 cache misses but hitting memory/disk .

CPU Resource and Memory Limiting

- Involuntary context switch occurs when a thread has been running too long without making a system call.

Workload (Status: Disk Sleep) → CPU IDLE Y% Memory X MB

Page Fault

Non voluntary context switch

Workload (Status: Disk Sleep) → CPU IDLE YY% Memory XX MB

Page Fault

Non voluntary context switch

A context switch can be voluntary - where a process either completes its task before its allotted time slice is over. Alternatively a context switch can be involuntary wherein if a process exhausts the cpu time slice allotted to it the kernel will pre-empt the process and switch it out to grant cpu time to another process in the run queue.

Workload (Status: Running) → CPU IDLE YYY% Memory XXX MB

Page Fault

Non voluntary context switch

Memory Limiting Tools

- Control Group (cgroup): It is a Linux kernel feature that allows to specify resources to a group of processes.
- CPU time, system memory, network bandwidth, or combinations of these resources can be set to the processes using cgroup.

Linux Commands:

```
service cgconfig restart
mkdir /cgroup/cpu_and_memory
mount -t cgroup -o memory cpu_and_memory  /cgroup/cpu_and_memory
cgcreate -g memory:/100mb

echo $(( 100 * 1024 * 1024 )) > /cgroup/cpu_and_mem/100mb/memory.limit_in_bytes
echo $(( 100* 1024 * 1024 )) >
/cgroup/cpu_and_mem/100mb/memory.memsw.limit_in_bytes

cgexec -g memory:100mb ./timeexecutioncal_swaptions.sh

cgclassify -g subsystems:path_to_cgroup <PID>
e.g: cgclassify -g memory:200mb 1701
```

CPU Limiting Tools

- Same configuration applies for CPU resources.

Linux Commands:

```
mkdir /cgroup/conf1-cpu
mount -t cgroup -o cpu conf1-cpu /cgroup/conf1-cpu
cgcreate -g cpu:/test-subgroup
cgset -r cpu.shares=512 cpulimited
cgset -r cpu.shares=1024 lesscpulimited
```

- Applying both CPU and memory limit on a process result in error. Linux forum/blog suggests that old hardware and Red Hat Linux versions are the cause.

  https://bugzilla.redhat.com/show_bug.cgi?id=612805

Memory Limit Tools

- Create several sections on the memory with different limit and attach those sections to the workload during execution.

```
[root@MICADLAB1 ~]# ls /cgroup/cpu_and_mem/
100mb   15mb    25mb    40mb    60mb                   memory.force_empty         memo
10mb    1mb     2mb     4mb     70mb                   memory.limit_in_bytes      memo
120mb   200mb   30mb    50mb    cgroup.procs           memory.max_usage_in_bytes  memo
150mb   20mb    3mb     5mb     memory.failcnt         memory.memsw.failcnt       memo
[root@MICADLAB1 ~]# ls /cgroup/cpu_and_mem/100mb/
cgroup.procs                     memory.move_charge_at_immigrate
memory.failcnt                   memory.soft_limit_in_bytes
memory.force_empty               memory.stat
memory.limit_in_bytes            memory.swappiness
memory.max_usage_in_bytes        memory.usage_in_bytes
memory.memsw.failcnt             memory.use_hierarchy
memory.memsw.limit_in_bytes      notify_on_release
memory.memsw.max_usage_in_bytes  tasks
memory.memsw.usage_in_bytes
[root@MICADLAB1 ~]# cat  /cgroup/cpu_and_mem/100mb/memory.limit_in_bytes
104857600
[root@MICADLAB1 ~]#
```

$$\frac{104857600}{1024 * 1024} = 100\ MB$$

Workload Execution – SPEC CPU

- Three workloads each from floating and integer type from SPEC CPU benchmark are run on core1 (cores 3 to 11 off) at 2.3 GHz and their completion time along with virtual memory details are measured.

Workload	VM Stack (KB)	VM EXE (KB)	VM Lib (KB)	VM PTE (KB)	VM Data (KB)	Completion Time(sec)
Gobmk(i)	152	1444	2216	56	26252	646
Hmmer(i)	88	284	2216	100	23816	643
Libquantum(i)	88	576	6924	160	65724	622
Gamess (f)	176	8636	3264	84	644264	1032
Gromacs(f)	88	936	3264	88	15548	686
Milc(f)	88	120	2216	156	672812	551

cat /proc/<PID>/status : Displays memory consumption as stack, data, library. static variables etc.

cat /proc/<PID>/maps : Displays page table content.

getconf PAGESIZE : Displays pagesize, 4KB for this machine . The selection of pagesize manually is only valid for i386 hardware.

Workload Execution – SPEC CPU

Workload	IPC	Page Fault (M/s)	Branch Ref (M/s)	Branch Miss (%)	Cache Ref	Cache Miss (M/s)
Gobmk(i)	1.196	0	518.797	7.273	4.948	0.407
Hmmer(i)	1.983	0	216.144	0.491	2.015	0.009
Libquantum (i)	1.615	0	651.05	0.005	70.046	23.674
Gamess (f)	2.071	0	468.561	1.142	1.038	0.003
Gromacs(f)	1.261	0	89.117	5.131	6.205	0.003
Milc(f)	0.927	0.004	114.42	0.034	45.107	38.528

perf stat –p <PID> : Displays the process's properties in terms of IPC, Page faults, branch miss, cache miss etc.

free –m : Displays the amount of free and used memory, buffer and swap.

sh -c "sync; echo 3 > /proc/sys/vm/drop_caches" : It cleanup the caches and memory.

To free pagecache: echo 1 > /proc/sys/vm/drop_caches
To free dentries and inodes: echo 2 > /proc/sys/vm/drop_caches
To free pagecache, dentries and inodes: echo 3 > /proc/sys/vm/drop_caches

Usefull Linux Tools

- Pmap –x <PID>   : Memory address space of a process.

```
[root@MICADLAB1 mic615codes]# pmap -x 19897
19897:   ../run_base_ref_gcc43-64bit.0141/libquantum_base.gcc43-64bit 1397 8
Address           Kbytes     RSS   Dirty Mode   Mapping
0000000000400000      40      36       0 r-x--  libquantum_base.gcc43-64bit
0000000000609000       4       4       4 rw---  libquantum_base.gcc43-64bit
0000000001917000     132     132     132 rw---    [ anon ]
00000036cc800000     128     104       0 r-x--  ld-2.12.so
00000036cca1f000       4       4       4 r----  ld-2.12.so
00000036cca20000       4       4       4 rw---  ld-2.12.so
00000036cca21000       4       4       4 rw---    [ anon ]
00000036cd000000    1564     292       0 r-x--  libc-2.12.so
00000036cd187000    2048       0       0 -----  libc-2.12.so
00000036cd387000      16      12       8 r----  libc-2.12.so
00000036cd38b000       4       4       4 rw---  libc-2.12.so
00000036cd38c000      20      12      12 rw---    [ anon ]
00000036cd400000     524      20       0 r-x--  libm-2.12.so
00000036cd483000    2044       0       0 -----  libm-2.12.so
00000036cd682000       4       4       4 r----  libm-2.12.so
00000036cd683000       4       4       4 rw---  libm-2.12.so
00007f3f33ac3000   32772   32772   32772 rw---    [ anon ]
00007f3f37a4a000   32784   32784   32784 rw---    [ anon ]
00007f3f39a62000       8       8       8 rw---    [ anon ]
00007fffb7c96000      84       8       8 rw---    [ stack ]
00007fffb7d0b000       4       4       0 r-x--    [ anon ]
ffffffffff600000       4       0       0 r-x--    [ anon ]
----------------  ------  ------  ------
total kB           72200   66212   65752
```

virtual memory = part in physical memory + part on disk

The part in physical memory is RSS.

Workload Under Memory Limit

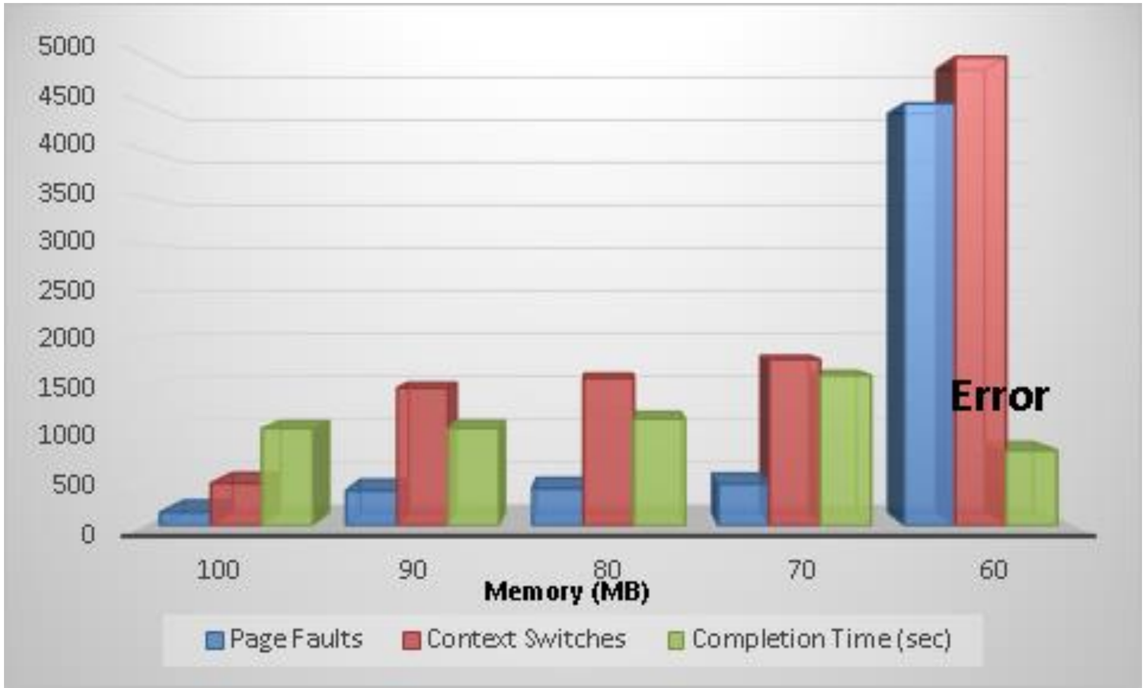
- Consider the workloads games (compute intensive).

Workload	VM Stack (KB)	VM EXE (KB)	VM Lib (KB)	VM PTE(KB)	VM Data (KB)
Gamess (f)	176	8636	3264	84	644264

- Minimum memory requirement = $stack + EXE + Lib + PTE$

Gamess: 11.875MB (Data: 629.16 MB)

- 1036 sec on 100 MB, 140 page faults, 450 context switches
- 1038 sec on 90 MB, 373 page faults, 1467 context switches
- 1139 sec on 80 MB, 394 page faults, 1,571 cont switches
- 1601 sec on 70 MB, 443 page faults, 1,771 cont switches
- Error on 60MB after 800sec, 4,497 page faults, 4,992 context switches



Workload Under Memory Limit

- Consider the workloads milc (memory intensive).

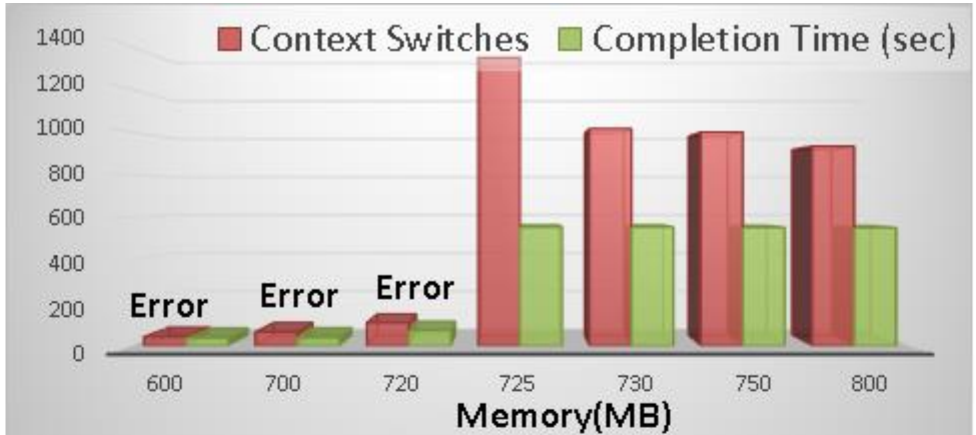
Workload	VM Stack (KB)	VM EXE (KB)	VM Lib (KB)	VM PTE(KB)	VM Data (KB)
Milc(f)	88	120	2216	156	672812

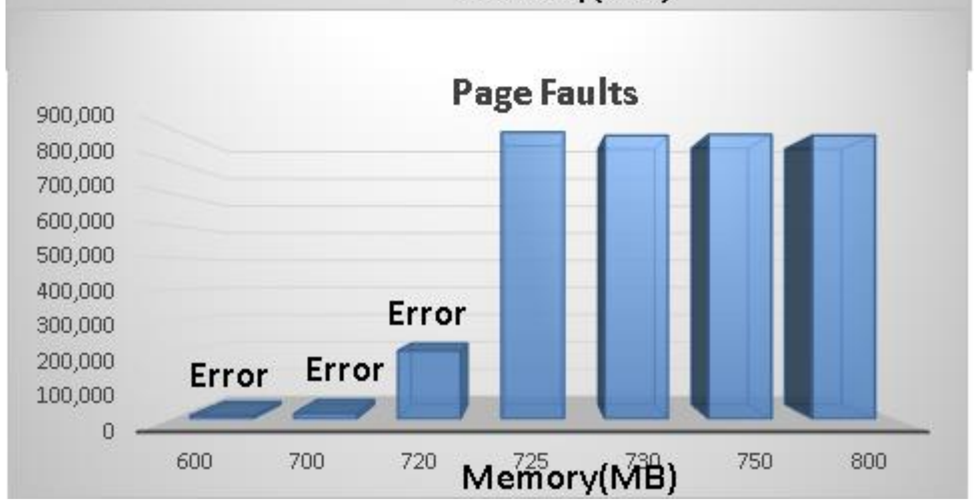
- Minimum memory requirement

$$= stack + EXE + Lib + PTE$$

= 2.519 MB (Data: 657.04 MB)

- Milc: Runtime error until 600MB.





Milc:
600 MB, 11,184 page faults, 42 context switches, 38 sec then error
700MB, 17,883 page faults, 65 context switches, 39 sec then error
720 MB, 211,552 page faults, 108 context switches, 72 sec then error.
725 MB, 888,762 page faults, 1,347 context switches, 555 sec
730 MB, 878,904 page faults, 1015, context switches, 555sec
750 MB, 881,723 Page faults, 999 context switches, 553sec
800 MB, 879,018 Page faults, 933 context switches, 552sec

Workload Under Memory Limit

- Consider the workloads libquantum (memory intensive).

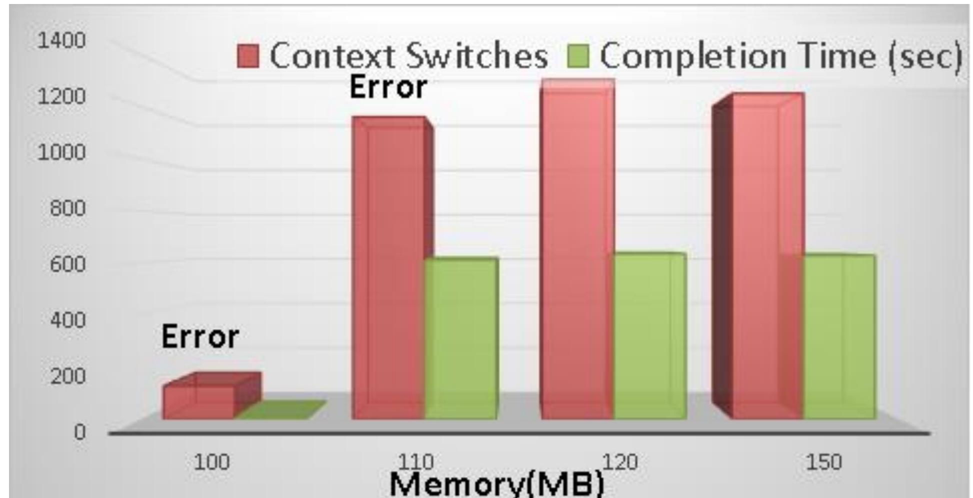
Workload	VM Stack (KB)	VM EXE (KB)	VM Lib (KB)	VM PTE(KB)	VM Data (KB)
Libquantum	88	576	6924	160	65724

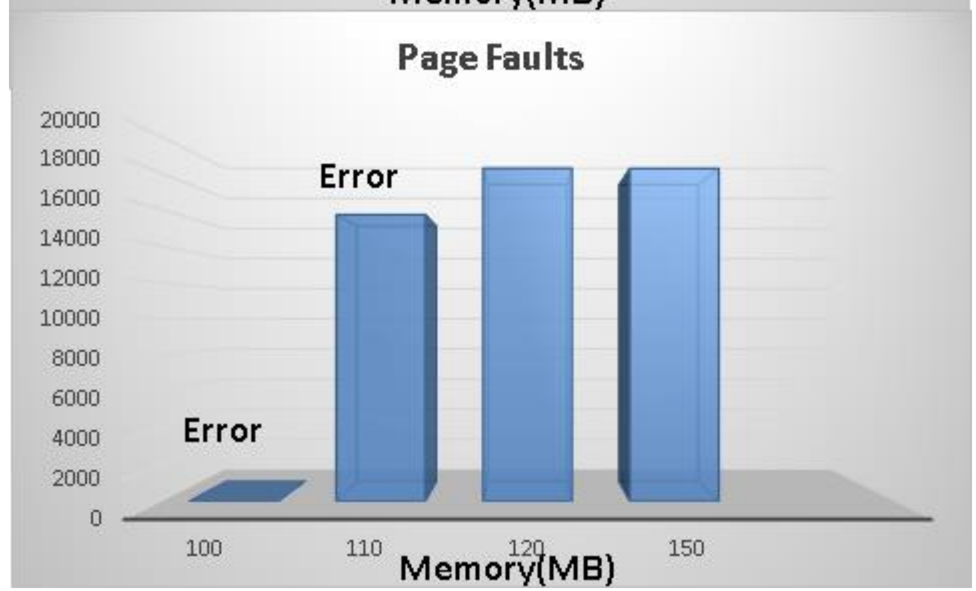
- Minimum memory requirement

$$= stack + EXE + Lib + PTE$$

= 7.566 MB (Data: 64.18 MB)

- Runtime error until 100 MB.





Libquantum:
150MB, 18,237 page faults, 1,251 context switches, 628sec
120 MB, 18,252 page faults, 1,305 context switches, 635 sec
110 MB, 15,709 page fault, 1161 context switches, 612 sec
100 MB, 0 Page faults, 127 context switches, 4 sec error.

Workload Execution - PARSEC

- Two workloads each from pipeline parallelization and data parallelization from PARSEC benchmark is run on core1 (cores 3 to 11 off) at 2.3 GHz.

Workload	VM Stack (KB)	VM EXE (KB)	VM Lib (KB)	VM PTE (KB)	VM Data (KB)	Completion Time(sec)
Bodytrack (p)	88	512	3324	92	155656	216
Ferret (p)	88	816	2308	108	480396	538
Canneal (d)	88	56	3324	692	497760	295
Freqmine (d)	88	100	3404	416	326480	661

Workload	IPC	Page Fault (M/s)	Branch Ref (M/s)	Branch Miss (%)	Cache Ref	Cache Miss (M/s)
Bodytrack (p)	1.750	0.001	5.234	0.525	5.234	0.067
Ferret (p)	1.342	0.001	528.588	4.418	5.635	2.500
Canneal (d)	0.538	0.001	142.271	1.218	15.775	10.796
Freqmine (d)	1.807	0	691.824	4.314	1.073	0.207

Workload Under Memory Limit

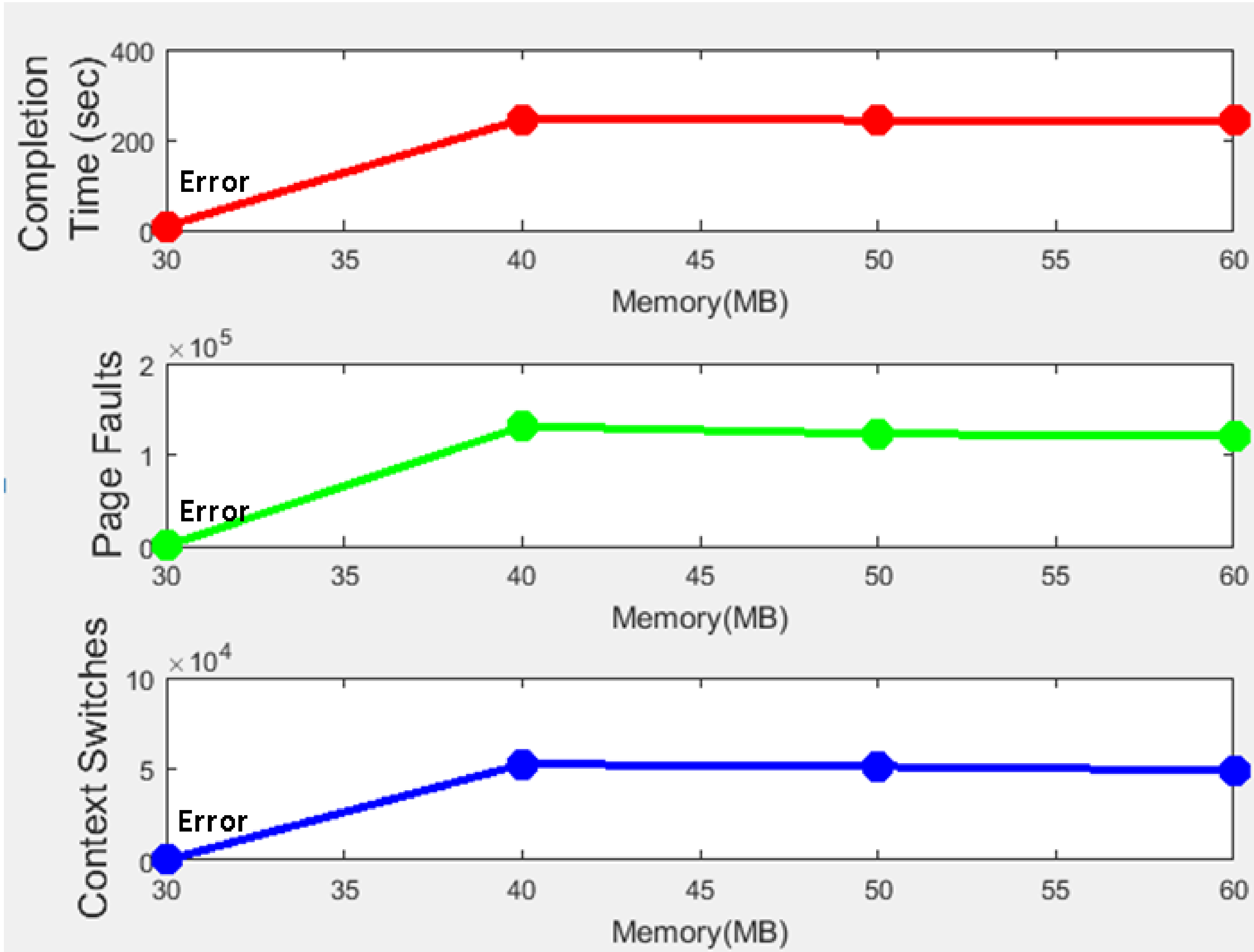
❖ Bodytrack minimum memory requirement $= stack + EXE + Lib + PTE$

= 3.92 MB

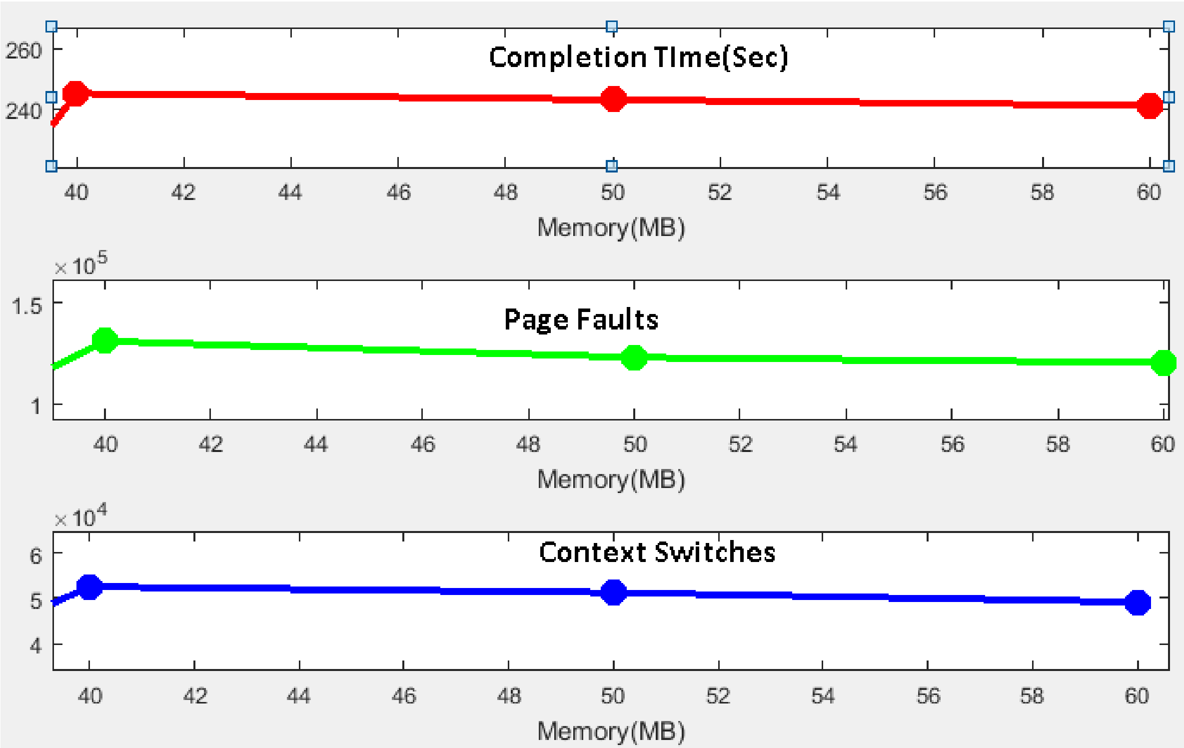
(Data: 152.01 MB)

Bodytrack:
50MB, 123,051 page faults, 51,231context switches, 243 sec
40MB, 131,078 page faults, 52,556 context switches, 245 sec
30MB, 38 sec error, no start
60MB, 120,470 Page Fault, 49,159 context switches, 241 sec

Canneal:
500 MB, 18,439 page faults, 151 context switches, 20 sec error
600 MB, 75,247 page faults, 382 context switches, 29 sec error
700 MB, 100,128 page faults, 534 context switches, 40 sec error
800 MB, 115,935 page faults, 691 context switches, 51 sec error



Workload Under Memory Limit



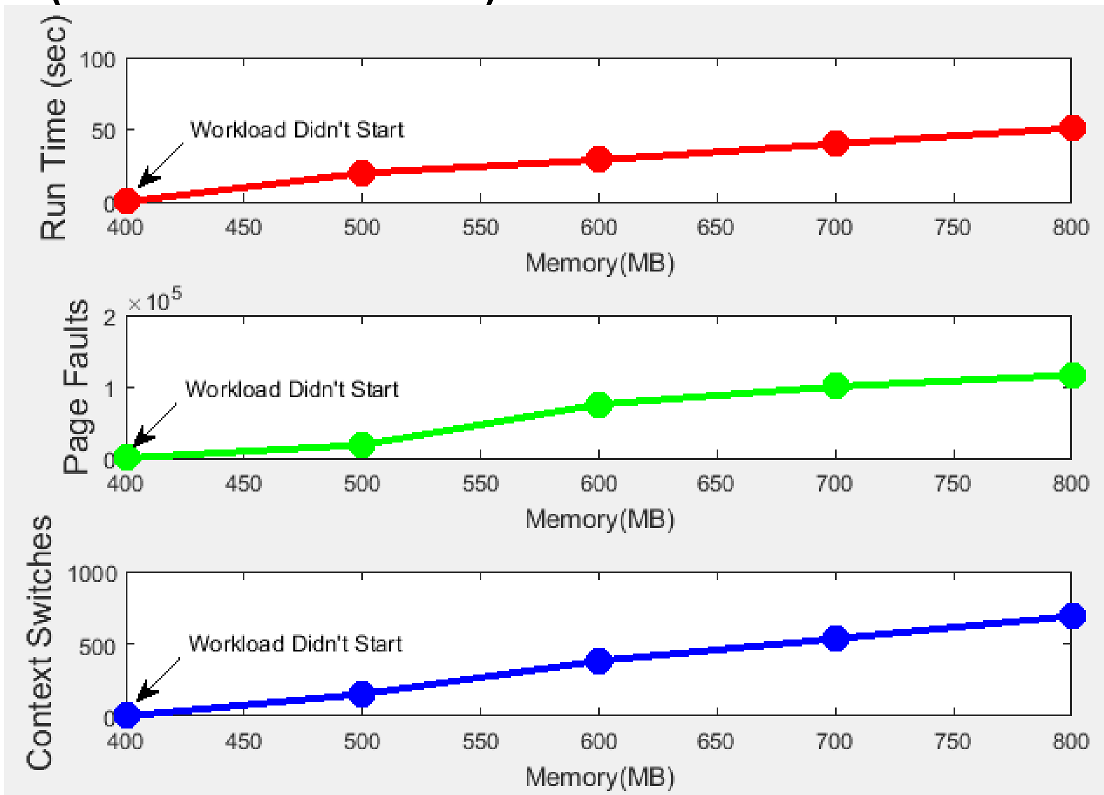
Bodytrack:
50MB, 123,051 page faults, 51,231context switches, 243 sec
40MB, 131,078 page faults, 52,556 context switches, 245 sec
30MB, 38 sec error, no start
60MB, 120,470 Page Fault, 49,159 context switches, 241 sec

Canneal:
500 MB, 18,439 page faults, 151 context switches, 20 sec error
600 MB, 75,247 page faults, 382 context switches, 29 sec error
700 MB, 100,128 page faults, 534 context switches, 40 sec error
800 MB, 115,935 page faults, 691 context switches, 51 sec error

Workload Under Memory Limit

- Minimum memory requirement = $stack + EXE + Lib + PTE$

  Canneal (Mem Int.) = 4.06 MB (Data: 486.09 MB)



Workload couldn't complete the execution.

Bodytrack:
50MB, 123,051 page faults, 51,231context switches, 243 sec
40MB, 131,078 page faults, 52,556 context switches, 245 sec
30MB, 38 sec error, no start
60MB, 120,470 Page Fault, 49,159 context switches, 241 sec

Canneal:
500 MB, 18,439 page faults, 151 context switches, 20 sec error
600 MB, 75,247 page faults, 382 context switches, 29 sec error
700 MB, 100,128 page faults, 534 context switches, 40 sec error
800 MB, 115,935 page faults, 691 context switches, 51 sec error

Workload Execution - Error

```
462.libquantum: copy 0 non-zero return code (exit code=0, signal=9)

Error: 1x462.libquantum
Producing Raw Reports
mach: default
  ext: gcc43-64bit
    size: ref
      set: int
NOTICE: sw_os001 is longer than 50 characters and will be split
        format: raw -> /opt/Installed_Softs/cpu2006/result/CINT2006.8
Parsing flags for 462.libquantum base: done
Doing flag reduction: done
        format: flags -> /opt/Installed_Softs/cpu2006/result/CINT2006
        format: ASCII -> /opt/Installed_Softs/cpu2006/result/CINT2006
        format: CSV -> /opt/Installed_Softs/cpu2006/result/CINT2006.8
        format: HTML -> /opt/Installed_Softs/cpu2006/result/CINT2006.
esult/CINT2006.8489.ref.gif
      set: fp

The log for this run is in /opt/Installed_Softs/cpu2006/result/CPU200
The debug log for this run is in /opt/Installed_Softs/cpu2006/result/

runspec finished at Wed May 10 16:26:57 2017; 1221 total seconds elap
It take 1221 sec to complete
```

Workload Execution - Success

```
Success: 1x462.libquantum
Producing Raw Reports
mach: default
  ext: gcc43-64bit
    size: ref
      set: int
NOTICE: sw_os001 is longer than 50 characters and will be split
        format: raw -> /opt/Installed_Softs/cpu2006/result/CINT
Parsing flags for 462.libquantum base: done
Doing flag reduction: done
        format: flags -> /opt/Installed_Softs/cpu2006/result/CI
        format: ASCII -> /opt/Installed_Softs/cpu2006/result/CI
        format: CSV -> /opt/Installed_Softs/cpu2006/result/CINT
        format: HTML -> /opt/Installed_Softs/cpu2006/result/CIN
esult/CINT2006.8490.ref.gif
      set: fp

The log for this run is in /opt/Installed_Softs/cpu2006/result/

runspec finished at Wed May 10 16:39:06 2017; 633 total seconds
It take 632 sec to complete
```

Workload Under CPU Limit

- Split the CPU resources using a 3:1 ratio.

```
[root@MICADLAB1 WorkloadSignature]# ls /cgroup/CPUMEM/
cgroup.procs   cpu.rt_period_us    cpu.shares       notify_on_release  tasks
cpulimited     cpu.rt_runtime_us  lesscpulimited  release_agent
[root@MICADLAB1 WorkloadSignature]#  cat /cgroup/CPUMEM/cpulimited/cpu.shares
250
[root@MICADLAB1 WorkloadSignature]# cat /cgroup/CPUMEM/lesscpulimited/cpu.shares
750
[root@MICADLAB1 WorkloadSignature]#
```

Workload	100% CPU Resource Completion Time(sec)	75 % CPU Resource Completion Time(sec)	25 % CPU Resource Completion Time(sec)
MILC (Mem Int.)	551	732	1581
Gamess (CPU Int.)	1032	1381	1577

- Memory intensive workload suffers performance degradation more than compute intensive workload by CPU resource sharing.

Workload Under CPU Limit

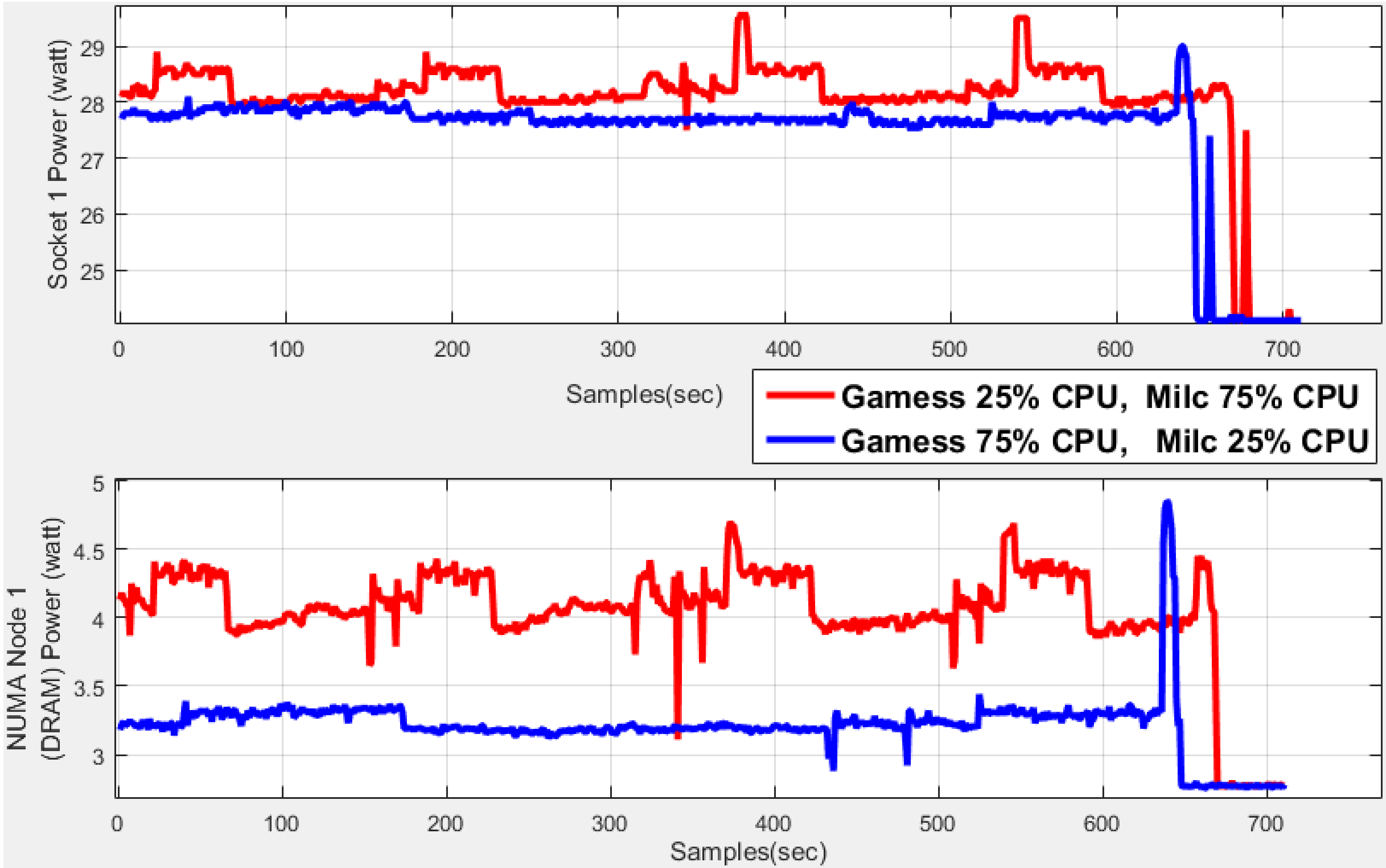
Cache Miss (M/s)			
Workload	**CPU Resource100%**	**CPU Resource75%**	**CPU Resource25%**
Milc	36.284	36.199	36.232
Gamess	0.003	0.101	0.292

Page Faults			
Workload	**CPU Resource100%**	**CPU Resource75%**	**CPU Resource25%**
Milc	880,421	883,378	882,108
Gamess	94	90	94

Context Switches			
Workload	**CPU Resource100%**	**CPU Resource75%**	**CPU Resource25%**
Milc	911	59,224	106,040
Gamess	358	50,303	60,103

- Cache miss rate and page faults remain approximately same under different CPU resource limit.

Power Comparisons



- Reducing the CPU resources for memory intensive workload reduces the processor and memory power consumption.

Machine Learning for Workload Sig.

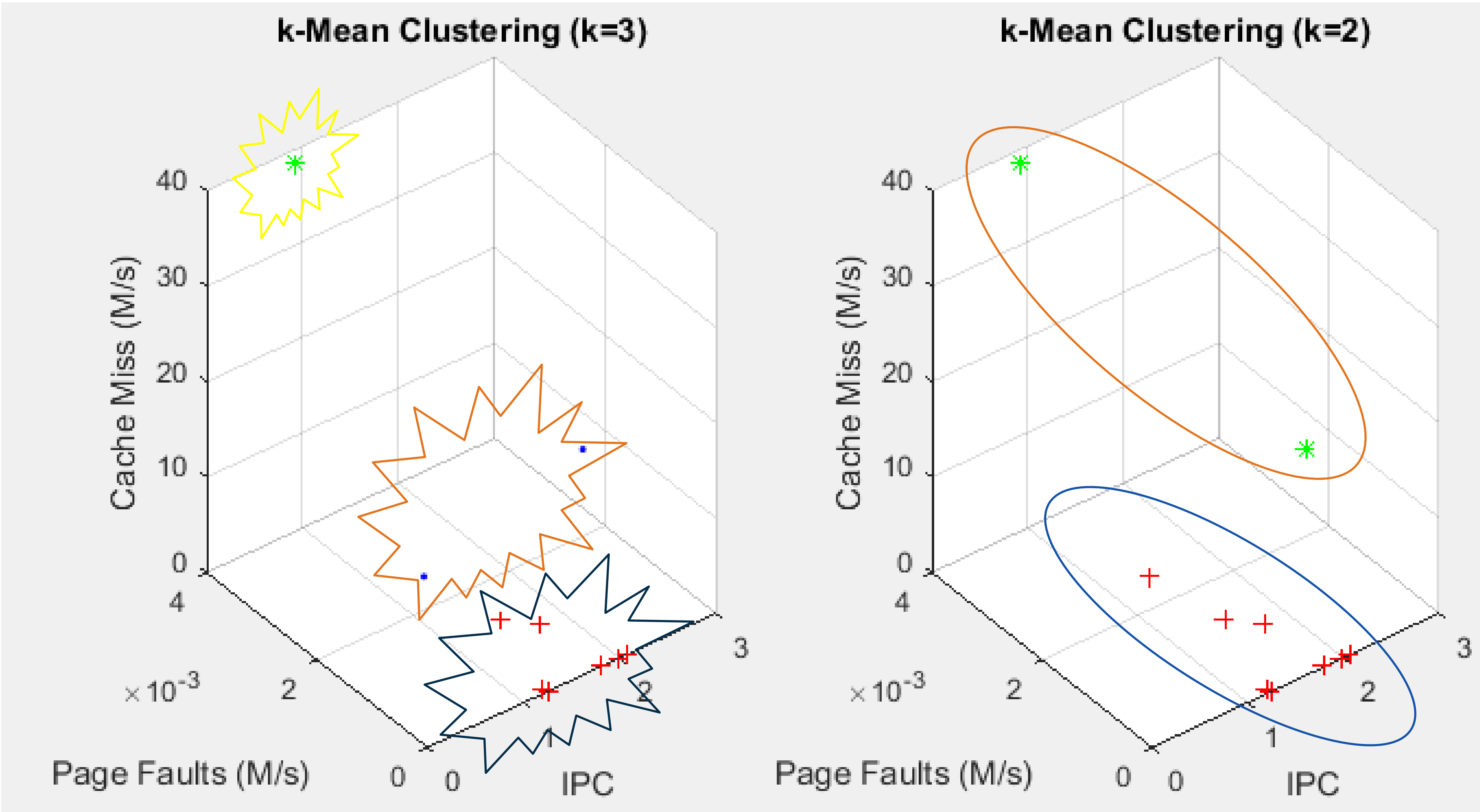
KN

```
>> KnnModel = fitcknn(X,Y,'NumNeighbors',2)

KnnModel =

  ClassificationKNN
           PredictorNames: {'x1'  'x2'  'x3'  'x4'  'x5'  'x6'}
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: [0 1]
           ScoreTransform: 'none'
          NumObservations: 6
                 Distance: 'euclidean'
             NumNeighbors: 2


  Properties, Methods

>> predict(KnnModel,[1.807  0.01    691.824 4.314   1.073   43.207])

ans =

     0
```

utput Label

IPC	Workload Sig
	0 – Compute
	0 – Compute
	1 – Memory
	0 – Compute
	0 – Compute
	1 – Memory

Machine Learning for Signature



Conclusion

- Workload's signature is calculated using Linux performance profiling tools. Possible use of machine learning for signature prediction is demonstrated.

- With cgroup, memory and CPU limits are exerted on the workload to measure the pagefaults, context switches, cache misses etc.

- Compute intensive workload needs comparatively small memory and has higher performance/throughput than memory intensive workload.

- Opensource Linux tools are explored. Those tools will help in memory scaling in cloud clusters (cloud-computing research).

- Could have done much better research if this course was provided in Masters.

SPEC – CPU Workload

416.gamess *(Floating point benchmarks cont'd)*

Author: Gordon Research Group, Iowa State University [1]

General Category: Quantum chemical computations

Description: A wide range of quantum chemical computations are possible using GAMESS. The benchmark 416.gamess does the following computations for the reference workload: (1) Self-consistent field (SCF) computation (type: Restricted Hartree-Fock) of cytosine molecule using the direct SCF method; (2) SCF computation (type: Restricted open-shell Hartee-Fock) of water and cu2+ using the direct SCF method; (3) SCF computation (type: Multi-configuration Self-consisted field) of triazolium ion using the direct SCF method

Inputs and Outputs: Described in the benchmark Docs subdirectory files INPUT.TXT, INTRO.TXT and PROG.TXT

Programming Language: Fortran

462.libquantum *(Integer benchmarks cont'd)*

Author: Björn Butscher, Hendrik Weimer

General Category: Physics / Quantum Computing

Description: libquantum is a library for the simulation of a quantum computer. Quantum computers are based on the principles of quantum mechanics and can solve certain computationally hard tasks in polynomial time.

In 1994, Peter Shor discovered a polynomial-time algorithm for the factorization of numbers, a problem of particular interest for cryptanalysis, as the widely used RSA cryptosystem depends on prime factorization being a problem only to be solvable in exponential time. An implementation of Shor's factorization algorithm is included in libquantum.

Libquantum provides a structure for representing a quantum register and some elementary gates. Measurements can be used to extract information from the system. Additionally, libquantum offers the simulation of decoherence, the most important obstacle in building practical quantum computers. It is thus not only possible to simulate any quantum algorithm, but also to develop quantum error correction algorithms. As libquantum allows to add new gates

Detailed Documentation at : https://www.spec.org/cpu2006/publications/CPU2006benchmarks.pdf

PARSEC Workload

Program	Application Domain	Parallelization
Blackscholes	Financial Analysis	Data-parallel
Bodytrack	Computer Vision	Pipeline NEW!
Canneal	Engineering	Data-parallel NEW!
Dedup	Enterprise Storage	Pipeline
Facesim	Animation	Data-parallel
Ferret	Similarity Search	Pipeline
Fluidanimate	Animation	Data-parallel
Freqmine	Data Mining	Data-parallel
Raytrace NEW!	Visualization	Data-parallel
Streamcluster	Data Mining	Data-parallel
Swaptions	Financial Analysis	Data-parallel
Vips	Media Processing	Data-parallel
X264	Media Processing	Pipeline

THANK YOU