Benchmark Report

HP ProLiant DL380 Gen9，Inspur NF5270M3，Sugon I620-G20

Version: 1.0.0

Date ： 2015

Agenda

- Benchmarking tools
- HP ProLiant DL380 Gen9
  - System Info
  - Pre-configuration
  - Test Report – STREAM
  - Test Report – Bandwidth
  - Test Report – pmbw
  - Test Report - Local Memory Access
  - Test Report - Remote Memory Access
  - Test Report – Lmbench
- Inspur NF5270M3
  - System Info
  - Pre-configuration
  - Test Report – STREAM
  - Test Report – Bandwidth
  - Test Report – pmbw
  - Test Report - Local Memory Access
  - Test Report - Remote Memory Access
  - Test Report – Lmbench
- HP ProLiant DL380 Gen9 vs. Inspur NF5270M3
- Sugon I620-G20
  - System Info
  - Pre-configuration
  - Test Report – STREAM
  - Test Report – Bandwidth
  - Test Report – pmbw
  - Test Report - Local Memory Access
  - Test Report - Remote Memory Access
  - Test Report – Lmbench

Benchmarking tools

- STREAM-5.10 http://www.cs.virginia.edu/stream/
  - for measuring sustained memory bandwidth
  - gcc -O -fopenmp stream.c -o stream_omp -DSTREAM_TYPE=double -DSTREAM_ARRAY_SIZE=256000000 -DOFFSET=56 -mcmodel=medium -DVERBOSE –DTUNED
- Bandwidth-1.1 http://zsmith.co/bandwidth.html
  - Bandwidth is a benchmark that attempts to measure memory bandwidth.
  - extended 'bandwidth' to measure network bandwidth as well.
- Pmbw-0.6.2(Parallel Memory Bandwidth Benchmark / Measurement) http://panthema.net/2013/pmbw/
  - to measure the parallel memory (cache and RAM) bandwidth of modern multi-core machines.
- Lmbench-3.0 http://www.bitmover.com/lmbench/
  - Bandwidth benchmarks
    - Cached file read
    - Memory copy (bcopy)
    - Memory read
    - Memory write
    - Pipe
    - TCP
  - Latency benchmarks
    - Context switching.
    - Networking: connection establishment, pipe, TCP, UDP, and RPC hot potato
    - File system creates and deletes.
    - Process creation.
    - Signal handling
    - System call overhead
    - Memory read latency
  - Miscellanious
    - Processor clock rate calculation

3

HP ProLiant DL380 Gen9

4

System Info

HP ProLiant DL380 Gen9

- CPU (http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon%20E5-2690%20v3.html)
  - 2 x Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
  - 2 x 12cores, 64bit
- Bus Speed
  - 9.6 GT/s QPI (4800 MHz)
  - 5 GT/s DMI
- Memory
  - L1:
    10 x 32 KB 8-way set associative instruction caches
    10 x 32 KB 8-way set associative data caches
  - L2:
    10 x 256 KB 8-way set associative caches
  - L3:
    30 MB 20-way set associative shared cache
  - 8 x HP 16GB 2Rx4 PC4-2133P-L Kit RAM

```
[root@localhost STREAM]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 0 size: 65411 MB
node 0 free: 62165 MB
node 1 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
node 1 size: 65536 MB
node 1 free: 62734 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

- OS: CentOS7
  - [root@localhost STREAM]# uname -o
  - GNU/Linux
  - [root@localhost STREAM]# uname -r
  - 3.10.0-229.el7.x86_64
  - root@localhost STREAM]# gcc –v
  - …
  - gcc version 4.8.3 20140911 (Red Hat 4.8.3-9) (GCC)

Pre-configuration

Option 1 **(Balanced)** :
2 cpu config using SmartMemory RDIMMs
Channel 1: 2 x (16GB DR, 0GB, 0GB)
Channel 2: 2 x (16GB DR, 0GB, 0GB)
Channel 3: 2 x (0GB, 0GB, 16GB DR)
Channel 4: 2 x (0GB, 0GB, 16GB DR)

Max memory speed* 2133 MHz **Select**

- 2 Processors to set scaling_governor == performance [i=0~23]
  - echo performance > /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor
- Optimized memory config recommended (2 nodes: 2x64GB on ch1, ch2, ch3 and ch4 per processor) http://h22195.www2.hp.com/DDR4memoryconfig/Home/Legal

Server memory positions

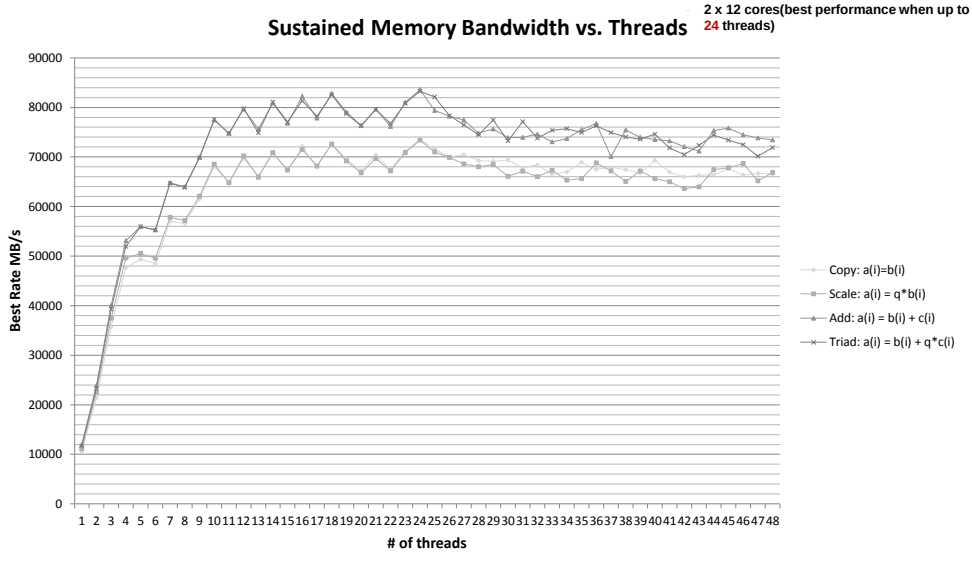
P1		P2	
1C	752369-081	1C	752369-081
2G	[empty slot]	2G	[empty slot]
3K	[empty slot]	3K	[empty slot]
4D	752369-081	4D	752369-081
5H	[empty slot]	5H	[empty slot]
6L	[empty slot]	6L	[empty slot]
7J	[empty slot]	7J	[empty slot]
8F	[empty slot]	8F	[empty slot]
9B	752369-081	9B	752369-081
10I	[empty slot]	10I	[empty slot]
11E	[empty slot]	11E	[empty slot]
12A	752369-081	12A	752369-081

Test Report – STREAM

#define STREAM_TYPE double

Why lines look like this?

- **# of Memory channel == 4 (bandwidth grows linearly until reached channel number restriction, e.g. From 10000 to 50000 MB/s, threads == 4)**
- **4 x16GB Memory/channel/Processor**
- **2 x 12 cores(best performance when up to 24 threads)**



Test Report – STREAM（cont.）

- The *Copy* benchmark measures the transfer rate in the absence of arithmetic. This should be one of the fastest memory operations, but it also represents a common one – fetching two values from memory, a(*i*) and b(*i*), and update one operation.
- The *Scale* benchmark adds a simple arithmetic operation to the Copy benchmark. This starts to simulate real application operations. The operation fetches two values from memory, a(*i*) and b(*i*), but operates on b(*i*) before writing it to a(*i*). It's a simple scalar operation, but more complex operations are built from it, so the performance of this simple test can be used as an indicator of the performance of more complex operations.
- The third benchmark, the S*um* benchmark, adds a third operand and was originally written to allow multiple load/store ports on vector machines to be tested when vector machines were in vogue. However, this benchmark is very useful today because of the large pipelines that some processors possess. Rather than just fetch two values from memory, this micro-benchmark fetches three. For larger arrays, this will quickly fill a processor pipeline, so you can test the memory bandwidth filling the processor pipeline or the performance when the pipeline is full. Moreover, this benchmark is starting to approximate what some applications will perform in real computations.
- The fourth benchmark in Stream, the T*riad* benchmark, allows chained or overlapped or fused, multiple-add operations. It builds on the Sum benchmark by adding an arithmetic operation to one of the fetched array values. Given that fused multiple-add operations (FMA) are an important operation in many basic computations, such as dot products, matrix multiplication, polynomial evaluations, Newton's method for evaluation functions, and many DSP operations, this benchmark can be directly associated with application performance. The FMA operation has its own instruction set now and is usually done in hardware. Consequently, feeding such hardware operations with data can be extremely important – hence, the usefulness of the Triad memory bandwidth benchmark.

Test Report – STREAM（cont.）

- STREAM log

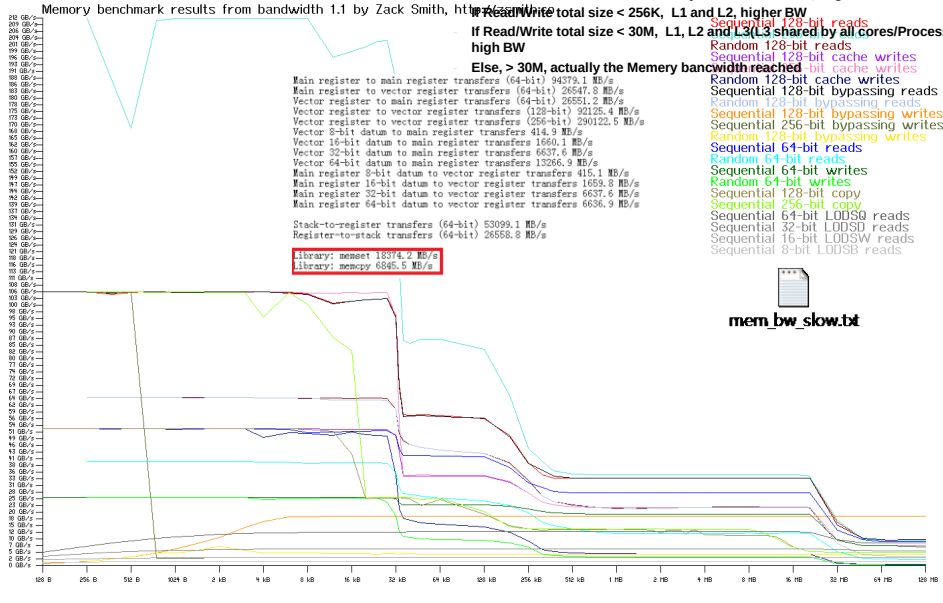
stream_omp-cpu_performance_enable.txt

Test Report – Bandwidth

Bandwidth --slow (high precision, ~8hrs)

Why lines look like this?

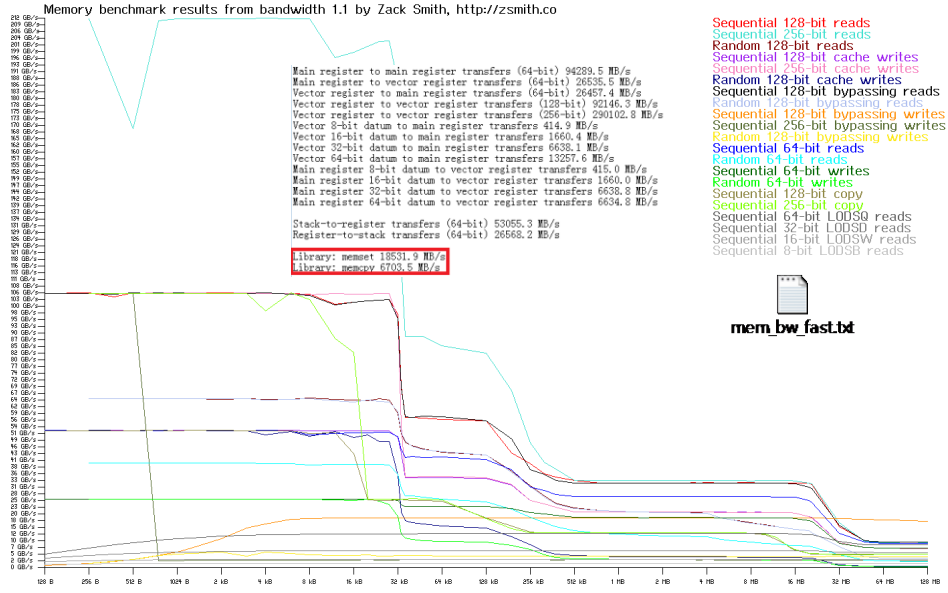
- **L1 cache = 32KB, L2 cache = 256K, L3 cache = 30M; cache line size = 64Byte(so 256bit access has best performence)**
- **X -> cache/Memory Size(B/KB/MB)**
- **Y-> Bandwidth GB/s**
- **If Read/Write total size < 32K, always access L1 cache, highest bandwidth**
- **If Read/Write total size < 256K, L1 and L2, higher BW**
- **If Read/Write total size < 30M, L1, L2 and L3(L3 shared by all cores/Processor), high BW**
- **Else, > 30M, actually the Memery bandwidth reached**



mem_bw_slow.txt

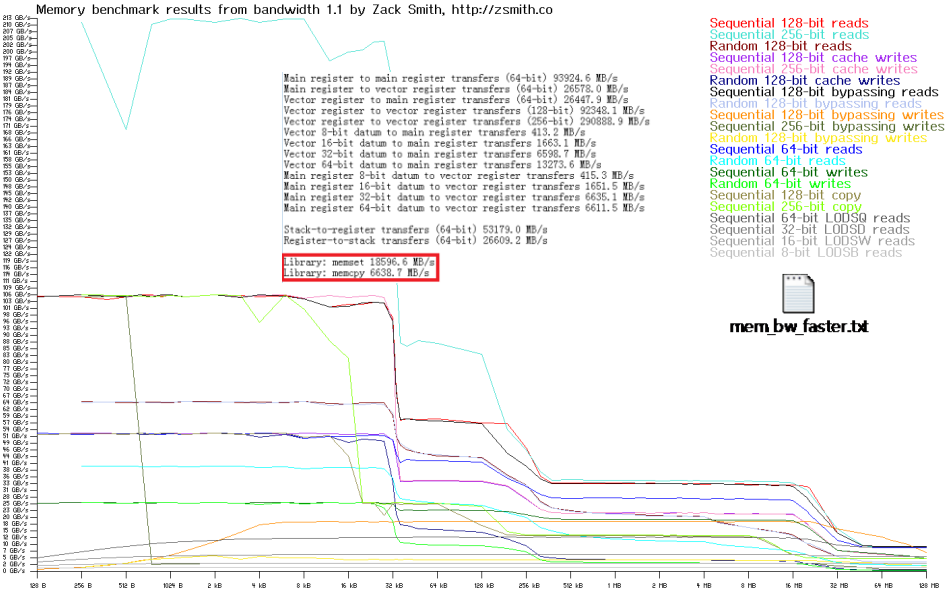
Test Report – Bandwidth(cont.)

Bandwidth --fast



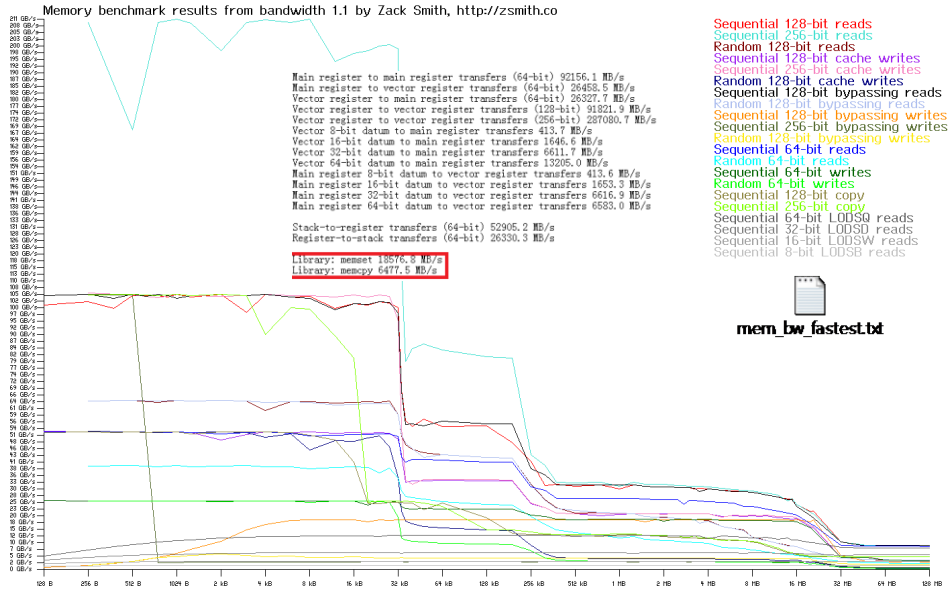
Test Report – Bandwidth(cont.)

Bandwidth --faster



Test Report – Bandwidth(cont.)

Bandwidth --fastest



Test Report – pmbw

Terminology

The names of the benchmark routines is composed of several abbreviated components, which together specify the exact operations:

- **Scan** indicates scanning operations, while **Perm** are permutation walking tests (see below).
- **Write/Read** specifies the operation done.
- **16/32/64/128/256** indicates the number of bits transfered by a single instruction in the benchmark routine. Which exactly are available depends on the architecture.
- **Ptr** represents pointer-based iteration, while **Index** is index-based array access.
- **SimpleLoop** routines contain only one operation per loop, after which the end condition is checked. **UnrollLoop** benchmarks contain 16 operations per loop, followed by the end check.

Test Report – pmbw(cont.)

Terminology

- Scanning Operation

The benchmark routines are coded in gcc inline assembly, however, for easier exposition, the same routines are shown in C/C++ syntax below. We first set `size_t an = n / sizeof(uint64_t);` to correctly count the number of 64-bit integers in the array.

A selection of the 64-bit scanning benchmarks in C/C++ representation are shown below. The first batch use pointer-based iteration, as would be used with STL iterators:

```
// ScanRead64PtrSimpleLoop
for (uint64_t* p = array; p < array + an; ++p)
    uint64_t x = *p;
```

```
// ScanWrite64PtrSimpleLoop
uint64_t x = 0xC0FFEE0C0FFEE00;
for (uint64_t* p = array; p < array + an; ++p)
    *p = x;
```

```
// ScanRead64PtrUnrollLoop
for (uint64_t* p = array; p < array + an; ++p) {
    uint64_t x0 = *(p+0);
    uint64_t x1 = *(p+1);
    // ... 13 times
    uint64_t x15 = *(p+15);
}
// ScanWrite64PtrUnrollLoop omitted
```

The following second batch of benchmark routines use array index-based access operations:

```
// ScanRead64IndexSimpleLoop
for (size_t i = 0; i < an; ++i)
    uint64_t x = array[i];
```

```
// ScanWrite64IndexSimpleLoop
uint64_t x = 0xC0FFEE0C0FFEE00;
for (size_t i = 0; i < an; ++i)
    array[i] = x;
```

```
// ScanRead64IndexUnrollLoop
for (size_t i = 0; i < an; i += 16) {
    uint64_t x0 = array[i+0];
    uint64_t x1 = array[i+1];
    // ... 13 times
    uint64_t x15 = array[i+15];
}
// ScanWrite64IndexUnrollLoop omitted
```

When using multiple threads, the scanned array is divided evenly among threads. The following figure illustrates the access pattern:

Test Report – pmbw(cont.)

Terminology

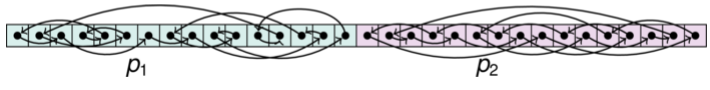
- Random Permutation Walking

Then the benchmark routine walks the permutation, it finishes when the cycle is complete.
The C/C++ representations of this benchmark's two routines is shown below:

```
// PermRead64SimpleLoop
uint64_t p = *array;
while( (uint64_t*)p != array )
    p = *(uint64_t*)p;
```

```
// PermRead64UnrollLoop
uint64_t p = *array;
while( (uint64_t*)p != array ) {
    p = *(uint64_t*)p;
    // ... 14 more times
    p = *(uint64_t*)p;
}
```

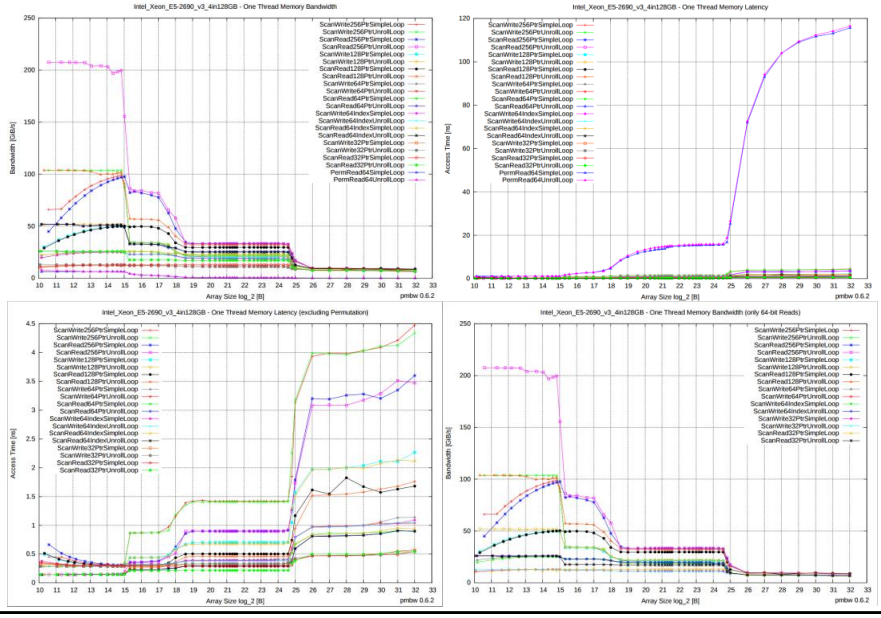
Before the test benchmark routine is called, the array area of each thread is filled with a random cyclic permutation of pointers. Thus for two threads, the allocated array will have the following layout:



Test Report – pmbw(cont.)

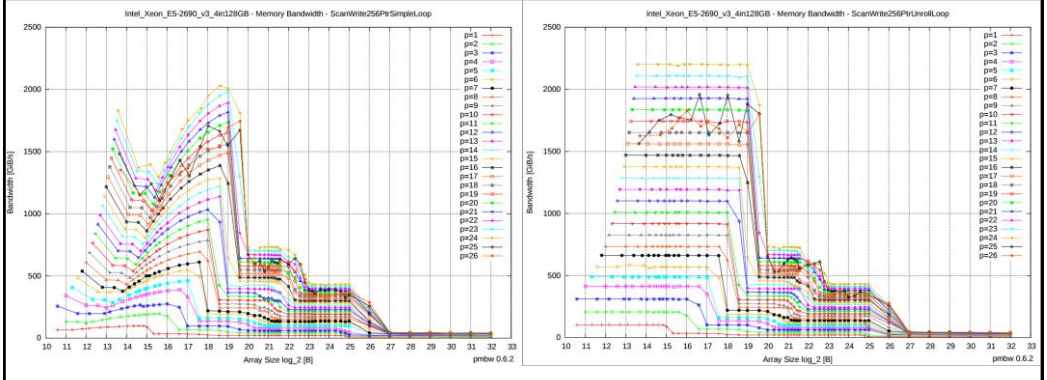
Why lines look like this?
- **Refer to Bandwidth above**

Single Thread Memory Bandwidth



Test Report – pmbw(cont.)

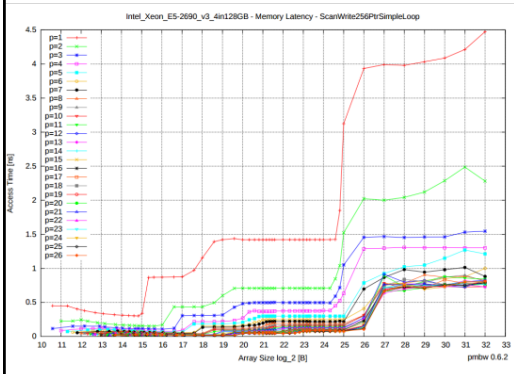
Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll

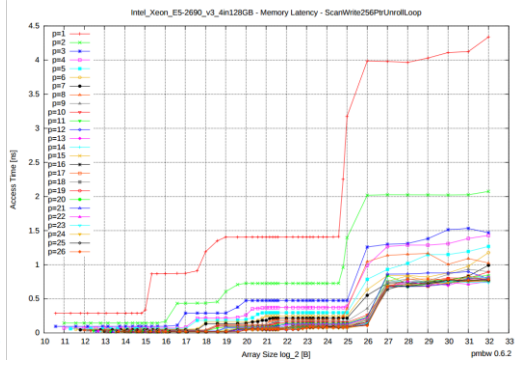


Why lines look like this?
- **P=24(parallel Cache/Memory access, 24 cores, 24 threads), so it shows best bandwidth/performance**
- **L1 = 32K, L2 = 256K, L3 = 30M(shared by all cores per Processor)**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll





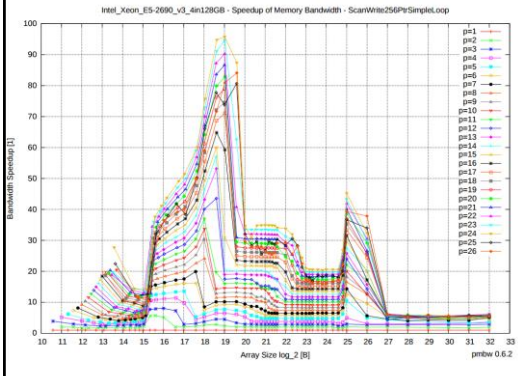
Why lines look like this?

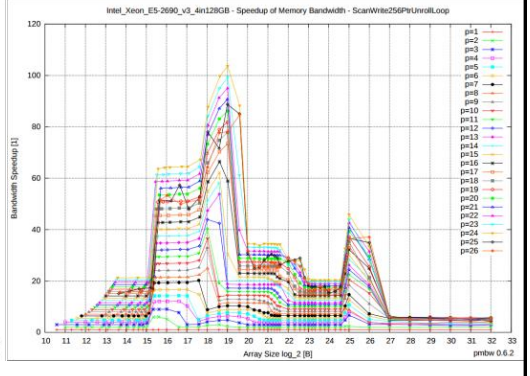
- **Refer to Bandwidth line chart on last slides, highest bandwidth has lowest Latency accordingly**

19

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll





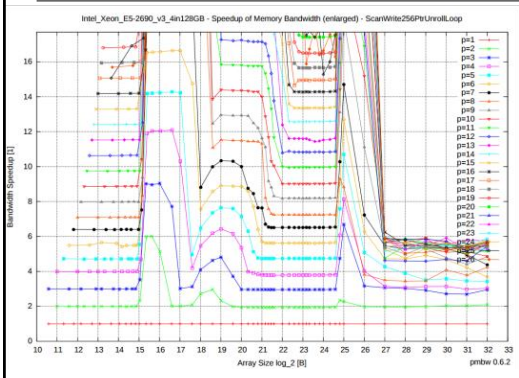
Why lines look like this?

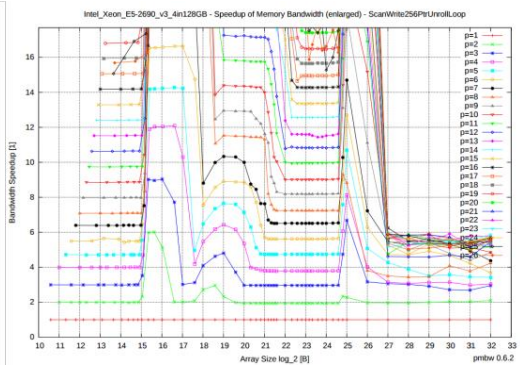
- **Speedup bandwidth depends on Parallel Memory Access(Multiple threads, maybe, have improved ratio of cache hit)**
- **Also it is restricted to the size of L1, L2 and L3**
- **Total 24 threads(one thread per core) have best speedup bandwidth**

20

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll





Test Report – pmbw(cont.)

Details for list below

ScanWrite256PtrSimpleLoop
ScanWrite256PtrUnrollLoop
ScanRead256PtrSimpleLoop
ScanRead256PtrUnrollLoop
ScanWrite128PtrSimpleLoop
ScanWrite128PtrUnrollLoop
ScanRead128PtrSimpleLoop
ScanRead128PtrUnrollLoop
ScanWrite64PtrSimpleLoop
ScanWrite64PtrUnrollLoop
ScanRead64PtrSimpleLoop
ScanRead64PtrUnrollLoop
ScanWrite64IndexSimpleLoop
ScanWrite64IndexUnrollLoop
ScanRead64IndexSimpleLoop
ScanRead64IndexUnrollLoop
ScanWrite32PtrSimpleLoop
ScanWrite32PtrUnrollLoop
ScanRead32PtrSimpleLoop
ScanRead32PtrUnrollLoop

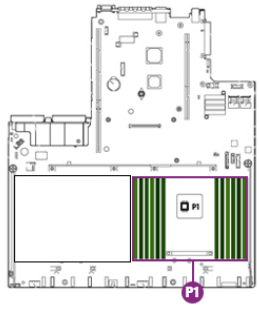
plots-Intel_Xeon_E5-2690_v3_4in128GB.pdf

Test Report - Local Memory Access

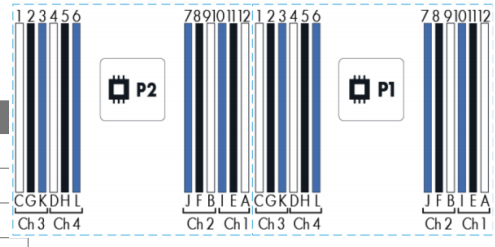
Memory layout

- 2x16GB on Processor1(ch1 and ch2)

Server memory positions.



P1	
1C	[empty slot]
2G	[empty slot]
3K	[empty slot]
4D	[empty slot]
5H	[empty slot]
6L	[empty slot]
7J	[empty slot]
8F	[empty slot]
9B	752371-081
10I	[empty slot]
11E	[empty slot]
12A	752371-081



What's Local Memory Access?

- For NUMA architecture, every processor has local memory
- Processor access its own memory only

Test Report - Local Memory Access(cont.)

CPU configure - Local

- Offline all Cores of Processor2 and enable scaling_governor ==“performance”for all cores of Processor1
  - echo 0 > /sys/devices/system/cpu/cpu#/online
  - echo performance > /sys/devices/system/cpu/cpu#/cpufreq/scaling_governor

```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
node 0 size: 32643 MB
node 0 free: 30079 MB
node distances:
node   0
  0:  10
[root@localhost ~]# cat /proc/cpuinfo | grep -E "processor|physical id"
processor       : 0
physical id     : 0
processor       : 1
physical id     : 0
processor       : 2
physical id     : 0
processor       : 3
physical id     : 0
processor       : 4
physical id     : 0
processor       : 5
physical id     : 0
processor       : 6
physical id     : 1
processor       : 7
physical id     : 1
processor       : 8
physical id     : 1
processor       : 9
physical id     : 1
processor       : 10
physical id     : 1
processor       : 11
physical id     : 1
processor       : 12
physical id     : 0
```

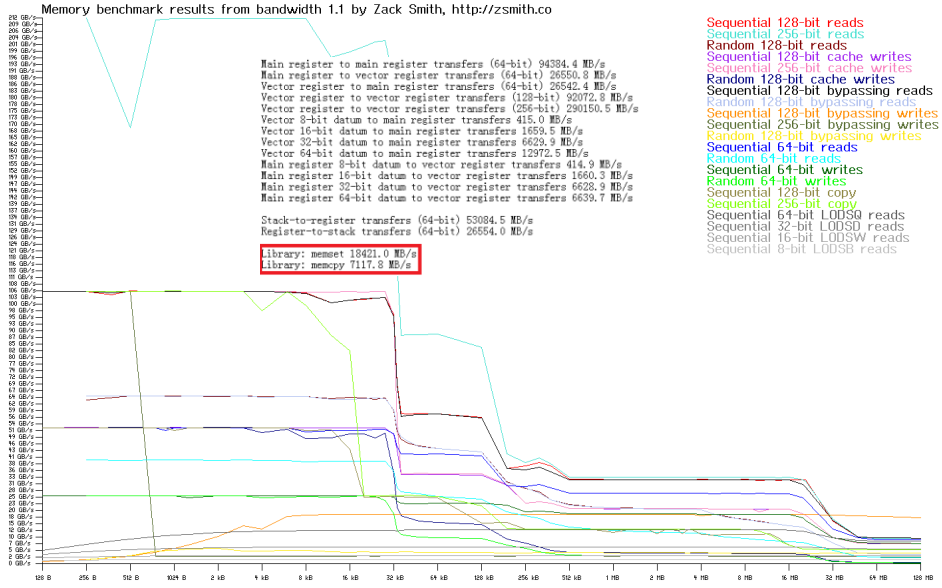
```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
node 0 size: 32643 MB
node 0 free: 30079 MB
node distances:
node   0
  0:  10
```

```
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu8/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu7/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu8/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu9/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu10/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu11/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu18/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu19/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu20/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu21/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu22/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu23/online
[root@localhost ~]#
```

```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 0 size: 32643 MB
node 0 free: 30084 MB
node distances:
node   0
  0:  10
```

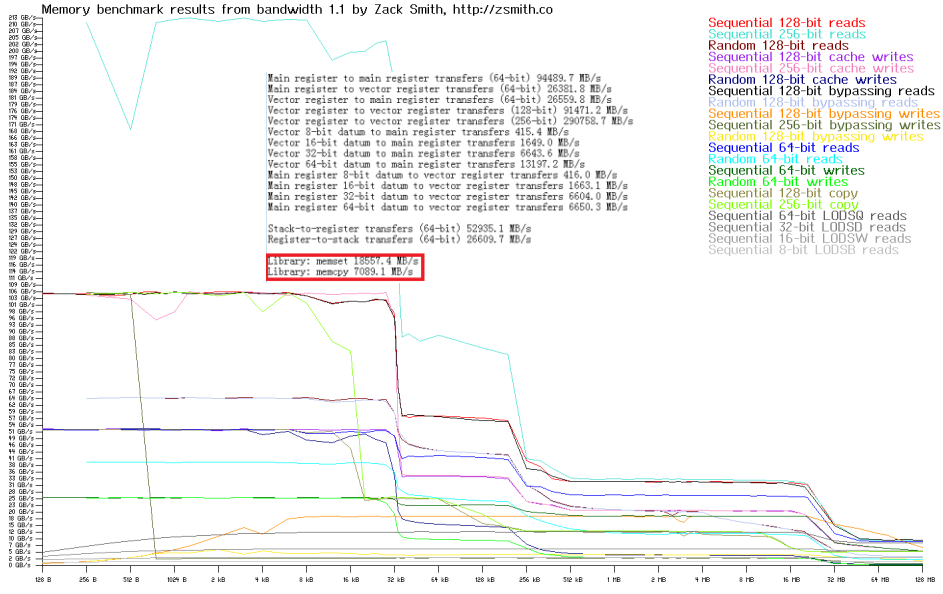
Test Report - Local Memory Access(cont.)

bandwidth --fast



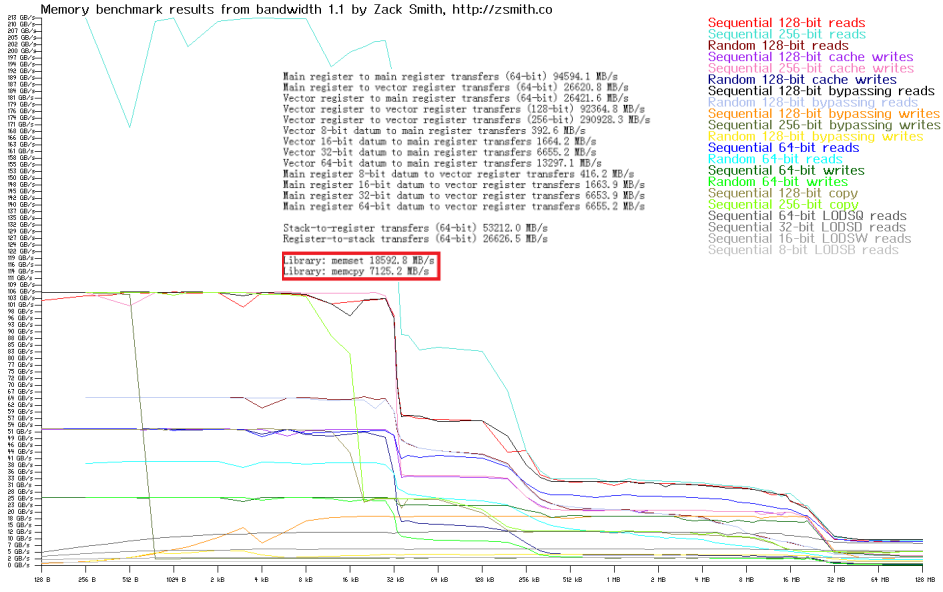
Test Report - Local Memory Access(cont.)

bandwidth --faster



Test Report - Local Memory Access

bandwidth --fastest



Test Report - Remote Memory Access via QPI

CPU configure - Remote

What's Remote Memory Access?

P2 access P1's Memory via QPI bus and vice versa.

- Offline all Cores of Processor1 and enable scaling_governor ==“performance”for all cores of Processor2
  - echo 0 > /sys/devices/system/cpu/cpu#/online
  - echo performance > /sys/devices/system/cpu/cpu#/cpufreq/scaling_governor

```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
node 0 size: 32643 MB
node 0 free: 30079 MB
node distances:
node   0
  0:  10
[root@localhost ~]# cat /proc/cpuinfo | grep -E "processor|physical id"
processor       : 0
physical id     : 0
processor       : 1
physical id     : 0
processor       : 2
physical id     : 0
processor       : 3
physical id     : 0
processor       : 4
physical id     : 0
processor       : 5
physical id     : 0
processor       : 6
physical id     : 1
processor       : 7
physical id     : 1
processor       : 8
physical id     : 1
processor       : 9
physical id     : 1
processor       : 10
physical id     : 1
processor       : 11
physical id     : 1
processor       : 12
physical id     : 0
```

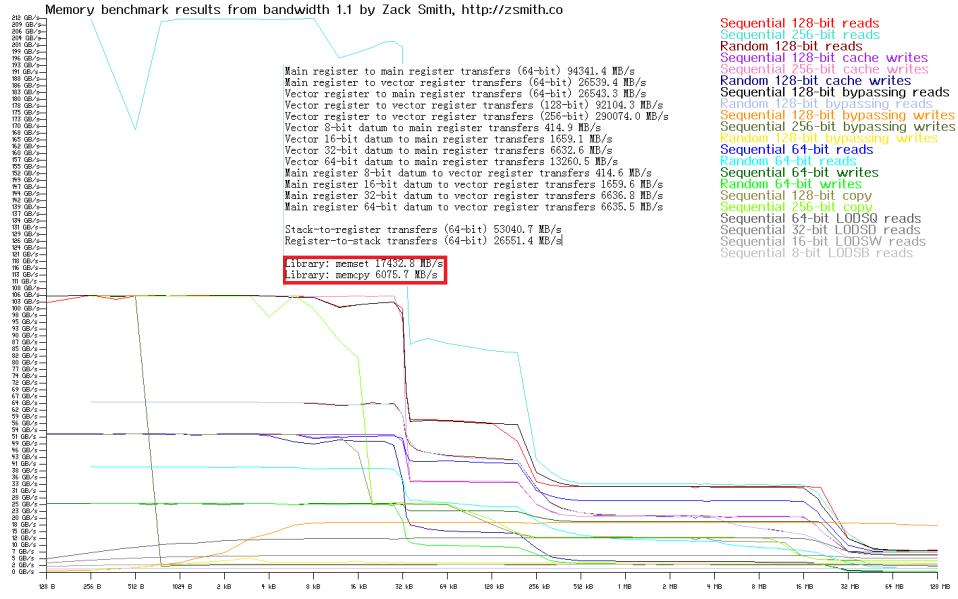
```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
node 0 size: 32643 MB
node 0 free: 30079 MB
node distances:
node   0
  0:  10
```

```
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu0/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu1/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu2/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu3/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu4/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu5/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu12/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu13/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu14/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu15/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu16/online
[root@localhost ~]# echo 0 > /sys/devices/system/cpu/cpu17/online
```

```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
node 0 size: 32643 MB
node 0 free: 30078 MB
node distances:
node   0
  0:  10
```

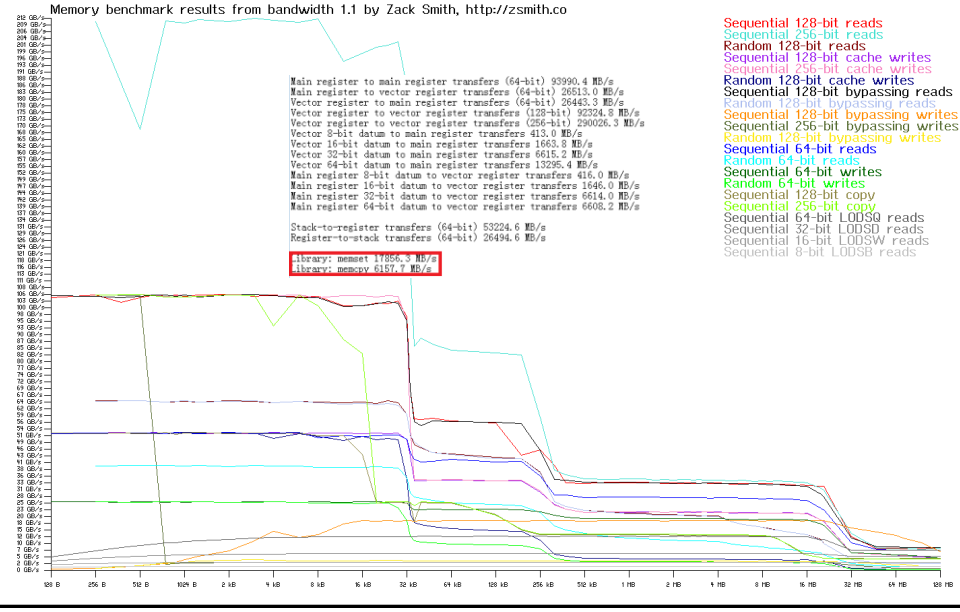
Test Report - Remote Memory Access via QPI(cont.)

bandwidth --fast



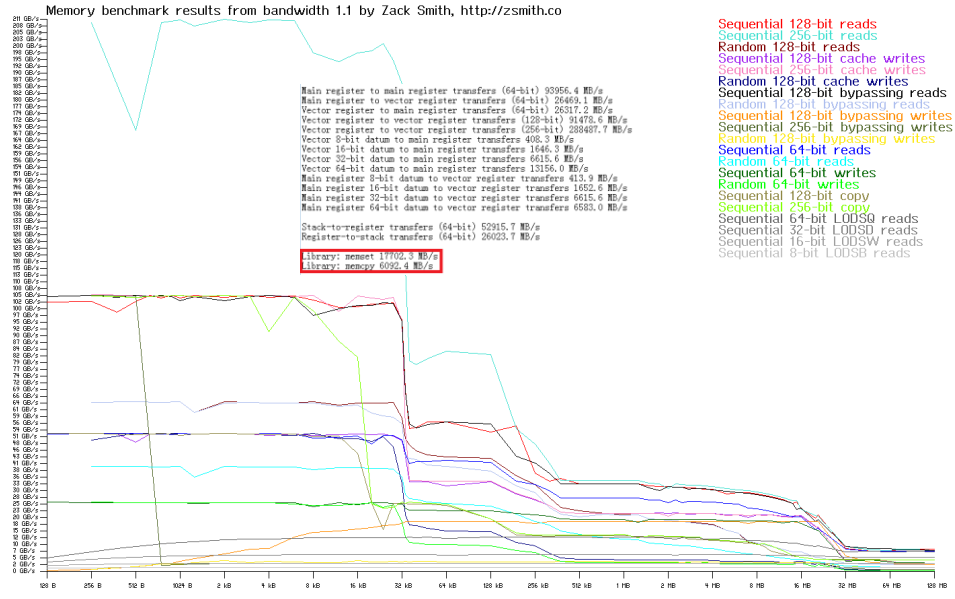
Test Report - Remote Memory Access via QPI(cont.)

bandwidth --faster



Test Report - Remote Memory Access via QPI(cont.)

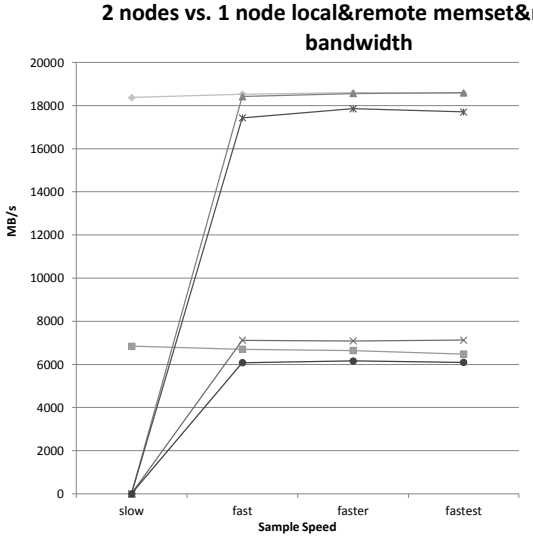
bandwidth --fastest



Test Report - Remote Memory Access via QPI(cont.)

2 nodes vs. 1 node local/remote

Memory Access	Memory Configuration	Activated Processor
Symmetric	2Nodes * 4x16GB DDR4 2133MHz	P0+P1
Local	1Node(#0) * 2x16GB DDR4 2133MHz	P0
Remote	1Node(#0) * 2x16GB DDR4 2133MHz	P1



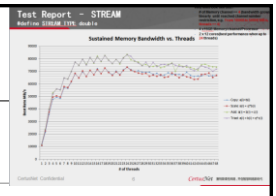
	2nodes memset	2nodes memcpy	1 node local memset	1 node local memcpy	1 node remote memset	1 node remote memcpy
slow	18374.2	6845.5	Not tested	Not tested	Not tested	Not tested
fast	18531.9	6703.5	18421	7117.8	17432.8	6075.7
faster	18596.6	6638.7	18557.4	7089.1	17856.3	6157.7
fastest	18576.8	6477.5	18592.8	7125.2	17702.3	6092.4

Why lines look like this?

- **In comparison with 2 nodes, 1 node local and 1 node remote memory access**
- **Memset always faster than memcpy**
- **Local memset/memcpy has highest performecnce, because of less memory access latency overhead**
- **Remote memory access has more overhead of QPI latency**

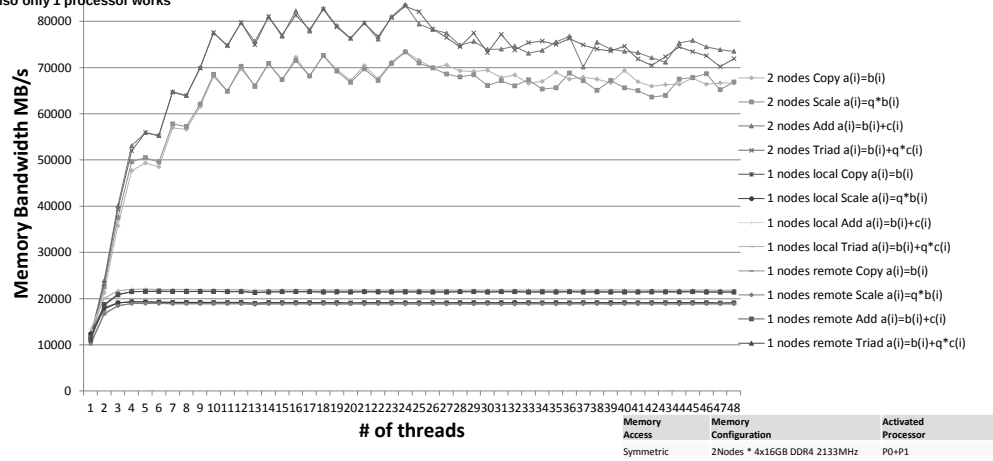
Test Report – STREAM

2 nodes vs. 1 node local vs. 1 node remote



Why lines look like this?

- 1 node * 2x16GB only resident 2 memory channels
- So 2 threads have best parallel memory performance, 4 threads reached threshold
- Also only 1 processor works



Memory Access	Memory Configuration	Activated Processor
Symmetric	2Nodes * 4x16GB DDR4 2133MHz	P0+P1
Local	1Node(#0) * 2x16GB DDR4 2133MHz	P0
Remote	1Node(#0) * 2x16GB DDR4 2133MHz	P1

33

Test Report – lmbench

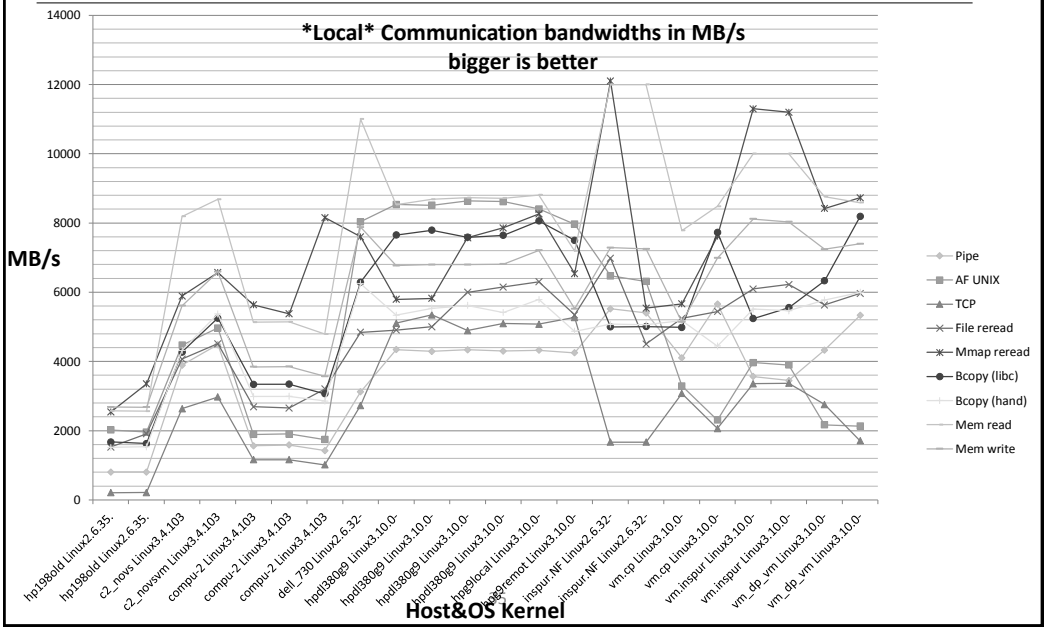
Server Info

Host		OS	Description	Processor/Memory
198	Linux	2.6.35.	old HP server(X8DTL?) + CentOS5.5	Intel Xeon E5506 @2.13GHz - 2x4cores
198	Linux	2.6.35.		
c2_novs	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch service stopped	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores DDR4 2133MHz
c2_novsvm	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch&VM service stopped	
compu-2	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0	
compu-2	Linux	3.4.103		
compu-2	Linux	3.4.103		
dell_730	Linux	2.6.32-	Dell PowerEdge R730 + CentOS6.6	
hpdl380g9	Linux	3.10.0-	HP ProLiant DL380 Gen9 + CentOS7	
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpg9local	Linux	3.10.0-		
hpg9remot	Linux	3.10.0-		
inspur.NF	Linux	2.6.32-	Inspur NF5270M3 + CentOS6.6	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
inspur.NF	Linux	2.6.32-		
vm.cp	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm.cp	Linux	3.10.0-		
vm.inspur	Linux	3.10.0	Inspur NF5270M3 + CentOS6.6 + KVM(CentOS7)	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
vm.inspur	Linux	3.10.0		
vm_dp_vm	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm_dp_vm	Linux	3.10.0-		

34

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others



Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Local Communication bandwidths in MB/s - bigger is better

Host	OS	Pipe	AF UNIX	TCP	File reread	Mmap reread	Bcopy (libc)	Bcopy (hand)	Mem read	Mem write
198	Linux 2.6.35-	804.	2019	209.	1526.5	2536.6	1671.9	1516.6	2573	2683.
198	Linux 2.6.35-	807.	1961	215.	1910.6	3351.4	1628.9	1533.9	2564	2676.
c2_novs	Linux 3.4.103	3894	4473	2638	4069.7	5890.2	4271.8	4318.1	8188	5611.
c2_novsvm	Linux 3.4.103	4484	4965	2974	4513.9	6568.7	5252.8	5375.5	8685	6587.
compu-2	Linux 3.4.103	1561	1891	1161	2689.9	5635.2	3340.7	2988.4	5134	3845.
compu-2	Linux 3.4.103	1585	1907	1161	2654.8	5377.2	3342.2	2992.2	5141	3853.
compu-2	Linux 3.4.103	1425	1742	1015	3203.4	8155.5	3066.8	2855.9	4789	3572.
dell_730	Linux 2.6.32-	3122	8037	2729	4842.8	7599.6	6291.0	6248.1	11.K	7880.
hpdl380g9	Linux 3.10.0-	4341	8534	5106	4905.4	5793.4	7651.4	5338.0	8529	6768.
hpdl380g9	Linux 3.10.0-	4291	8515	5347	5002.2	5819.7	7793.7	5533.8	8684	6798.
hpdl380g9	Linux 3.10.0-	4336	8638	4894	5997.2	7573.1	7588.6	5617.5	8731	6797.
hpdl380g9	Linux 3.10.0-	4302	8620	5094	6150.6	7861.3	7644.9	5412.7	8716	6810.
hpg9local	Linux 3.10.0-	4325	8405	5079	6300.7	8262.5	8057.9	5787.3	8808	7209.
hpg9remot	Linux 3.10.0-	4251	7968	5271	5345.4	6539.8	7499.9	4868.2	7191	5521.
inspur.NF	Linux 2.6.32-	5521	6474	1669	6983.1	12.1K	5000.9	5078.6	12.K	7288.
inspur.NF	Linux 2.6.32-	5402	6313	1670	4502.6	5537.6	5008.5	5063.6	12.K	7249.
vm.cp	Linux 3.10.0-	4106	3293	3077	5246.9	5665.9	4983.9	5180.3	7782	6138.
vm.cp	Linux 3.10.0-	5661	2308	2063	5443.1	7614.4	7732.3	4445.2	8482	6985.
vm.inspur	Linux 3.10.0	3566	3969	3356	6096.4	11.3K	5240.2	5511.9	10.K	8114.
vm.inspur	Linux 3.10.0	3449	3896	3373	6223.4	11.2K	5555.2	5465.9	10.K	8030.
vm_dp_vm	Linux 3.10.0-	4326	2171	2755	5633.8	8421.4	6328.3	5774.4	8762	7239.
vm_dp_vm	Linux 3.10.0-	5333	2131	1706	5966.7	8730.6	8187.2	5996.7	8587	7397.

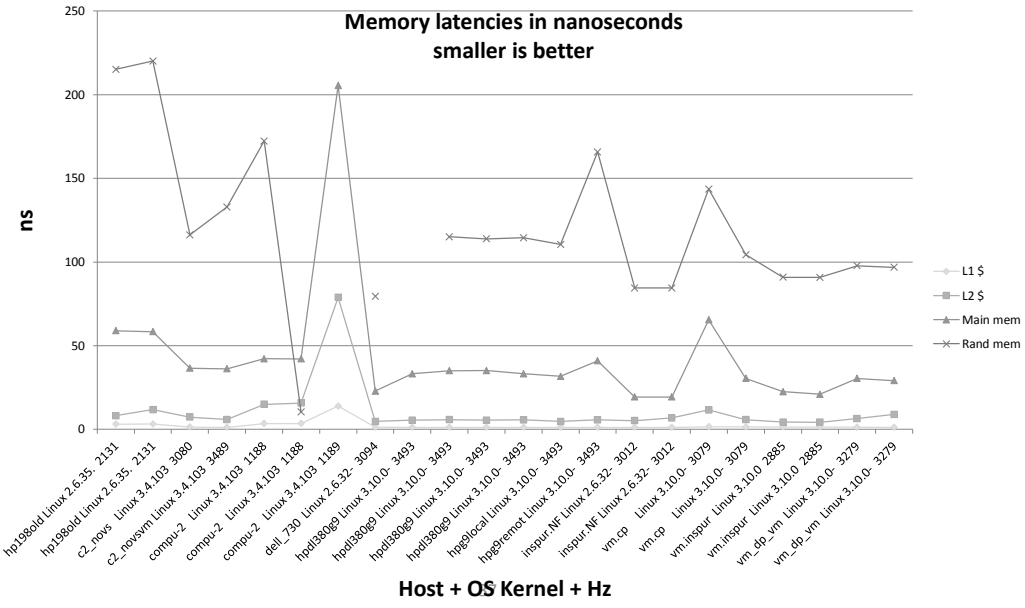
According to the test results, focus on performance on host machine

- **HP ProLiant DL380 Gen9(DDR4 2Rx4 2133MHz) has low performance of memory bandwidth, particularly, in Memory Read&Write vs. Inspur (DDR3 2Rx2 1333MHz)or Dell PowerEdge R730, the gap is up to ~3GB/s on read and ~0.5GB/s on write**
- **Remote memory access has lower performance, ~ 1.5GB/s on read and ~1.3GB/s on write**
- **Local memory access has a little better performance than 2 nodes**

Note: please ignore the error data caused by the instability of system environment !

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others



Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Memory latencies in nanoseconds - smaller is better
(WARNING - may not be correct, check graphs)

Host	OS	Mhz	L1 $	L2 $	Main mem	Rand mem	Guesses
198	Linux 2.6.35.	2131	3.0850	8.1480	58.9	215.1	
198	Linux 2.6.35.	2131	3.1910	11.8	58.4	220.1	
c2_novs	Linux 3.4.103	3080	1.3200	7.3290	36.6	116.2	
c2_novsvm	Linux 3.4.103	3489	1.1700	5.8940	36.2	132.9	
compu-2	Linux 3.4.103	1188	3.4640	14.9	42.2	172.3	
compu-2	Linux 3.4.103	1188	3.4420	15.7	42.1	10.4	
compu-2	Linux 3.4.103	1189	13.9	78.8	205.6		
dell_730	Linux 2.6.32-	3094	1.2930	4.6690	22.9	79.5	
hpdl380g9	Linux 3.10.0-	3493	1.1460	5.4760	33.2		
hpdl380g9	Linux 3.10.0-	3493	1.1470	5.7150	35.0	115.1	
hpdl380g9	Linux 3.10.0-	3493	1.1460	5.4980	35.1	113.8	
hpdl380g9	Linux 3.10.0-	3493	1.1470	5.7040	33.2	114.5	
hpg9local	Linux 3.10.0-	3493	1.1460	4.6780	31.7	110.5	
hpg9remot	Linux 3.10.0-	3493	1.1450	5.6430	41.0	165.8	
inspur.NF	Linux 2.6.32-	3012	1.2130	5.2230	19.3	84.5	
inspur.NF	Linux 2.6.32-	3012	1.2130	6.8120	19.3	84.5	
vm.cp	Linux 3.10.0-	3079	1.5880	11.6	65.4	143.6	
vm.cp	Linux 3.10.0-	3079	1.4700	5.7840	30.4	104.4	
vm.inspur	Linux 3.10.0	2885	1.3890	4.3220	22.5	90.9	
vm.inspur	Linux 3.10.0	2885	1.3910	4.2290	21.0	90.8	
vm_dp_vm	Linux 3.10.0-	3279	1.2260	6.4110	30.4	97.8	
vm_dp_vm	Linux 3.10.0-	3279	1.1520	8.8750	29.1	96.8	

For memoey latencies,

- **HP ProLiant DL380 Gen9(DDR4 2Rx4 2133MHz) has less latency than Inspur/Dell PowerEdge R730 on L1 Cache**
- **But HP has obvious more latency overhead vs Inspur/Dell on both sequential and random access on Main Memory**

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Micro-benchmarking Details

Inspur NF5270M3

System Info

Inspur NF5270M3

- CPU (http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon%20E5-2690%20v2.html)
  - 2 x Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
  - 2 x 10cores, 64bit
- Bus Speed
  - 8 GT/s QPI (4000 MHz)
  - 5 GT/s DMI
- Memory
  - L1:
    10 x 32 KB 8-way set associative instruction caches
    10 x 32 KB 8-way set associative data caches
  - L2:
    10 x 256 KB 8-way set associative caches
  - L3:
    25 MB 20-way set associative shared cache
  - 8 x 16GB 2Rx4 PC3L-12800R-11-13-E2 RAM(DDR3 1600MHz)

```
[root@localhost perf_tools]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9
node 0 size: 65474 MB
node 0 free: 63333 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19
node 1 size: 65536 MB
node 1 free: 63544 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

- OS: CentOS7
  - [root@localhost STREAM]# uname -o
  - GNU/Linux
  - [root@localhost STREAM]# uname -r
  - 3.10.0-229.el7.x86_64
  - root@localhost STREAM]# gcc –v
  - …
  - gcc version 4.8.3 20140911 (Red Hat 4.8.3-9) (GCC)

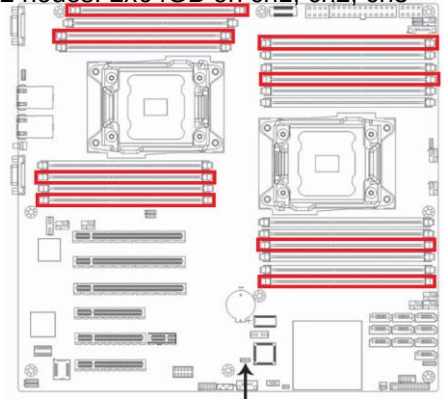
41

Pre-configuration

- 2 Processors to set scaling_governor == performance [i=0~19]

  echo performance > /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor

- Optimized memory config recommended (2 nodes: 2x64GB on ch1, ch2, ch3 and ch4 per processor)



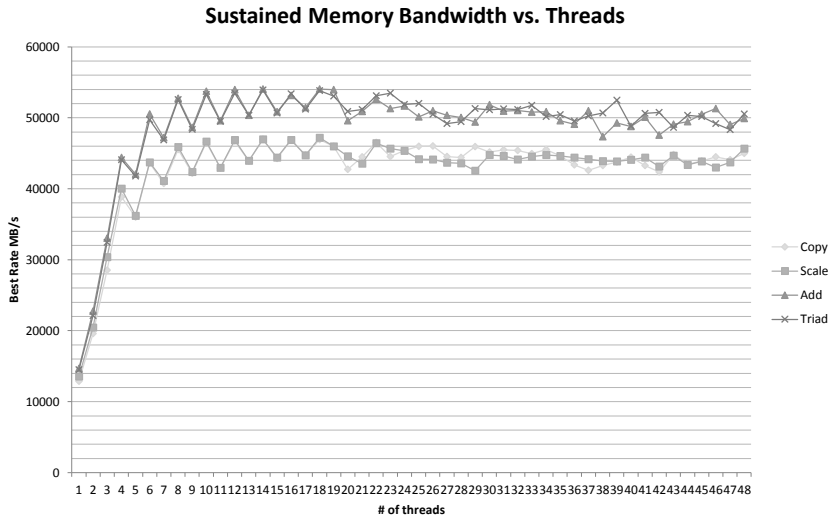
42

Test Report – STREAM

#define STREAM_TYPE double

Why lines look like this?

- **# of Memory channel == 4 (bandwidth grows linearly until reached channel number restriction, e.g. From 10000 to 50000 MB/s, threads == 4)**
- **4 x16GB Memory/channel/Processor**
- **2 x 12 cores(best performance when up to 24 threads)**



Test Report – STREAM（cont.）

- STREAM log

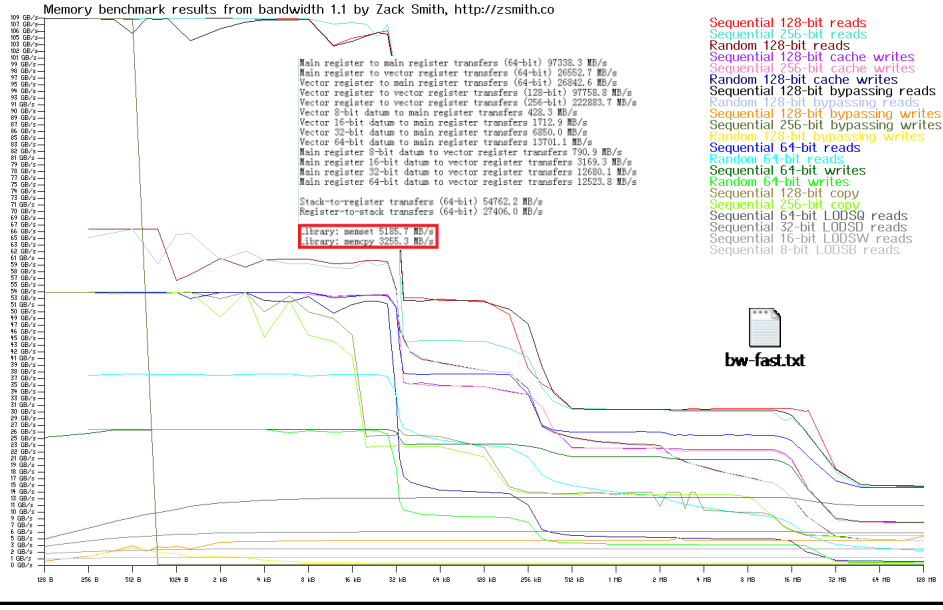
Test Report – Bandwidth

Bandwidth --slow (high precision, ~8hrs)

- Not Tested

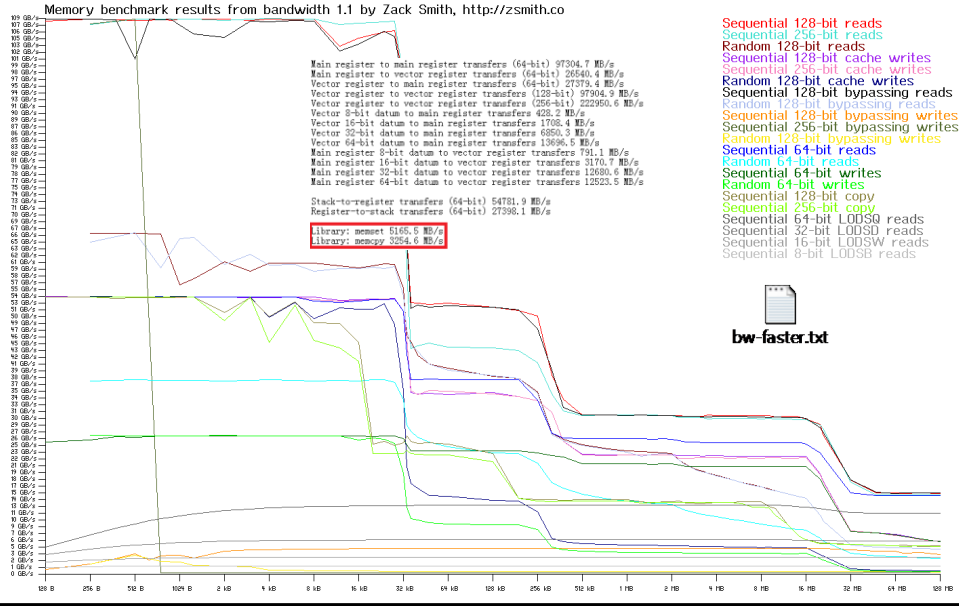
Test Report – Bandwidth(cont.)

Bandwidth --fast



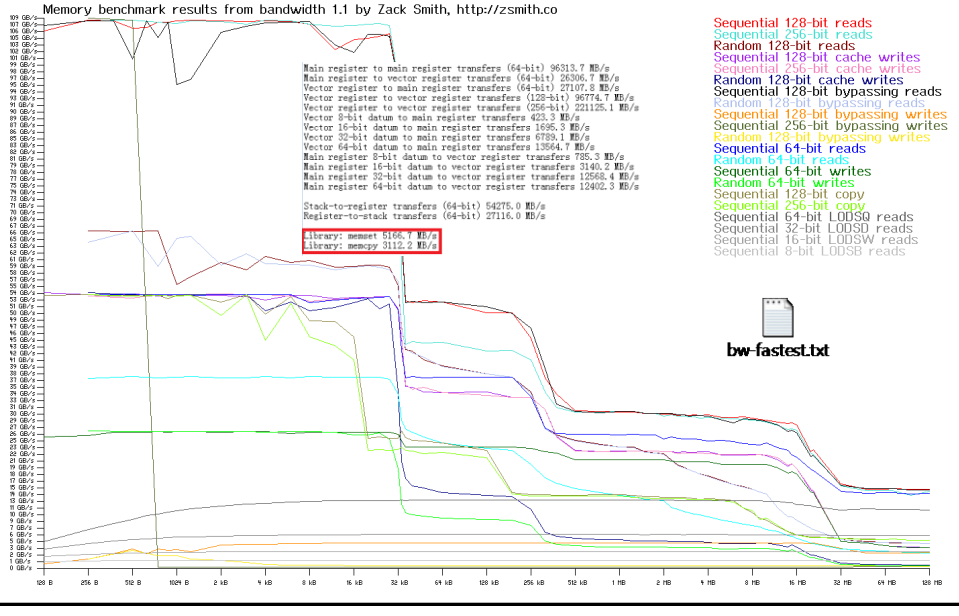
Test Report – Bandwidth(cont.)

Bandwidth --faster



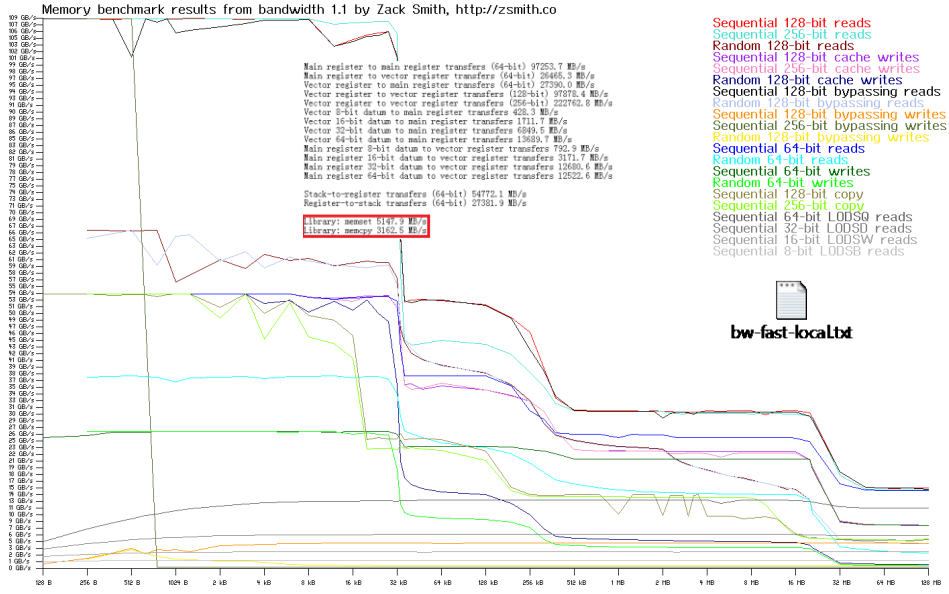
Test Report – Bandwidth(cont.)

Bandwidth --fastest



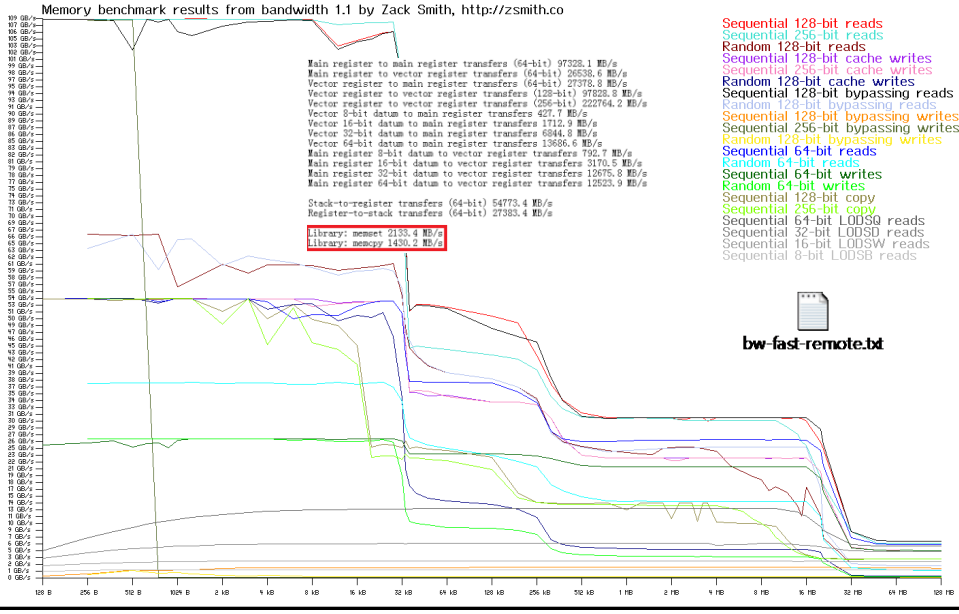
Test Report – Local Memory Access

Bandwidth –fast



Test Report – Remote Memory Access via QPI

Bandwidth --fast



Test Report – Bandwidth

2Nodes vs 1Node Remote/Local

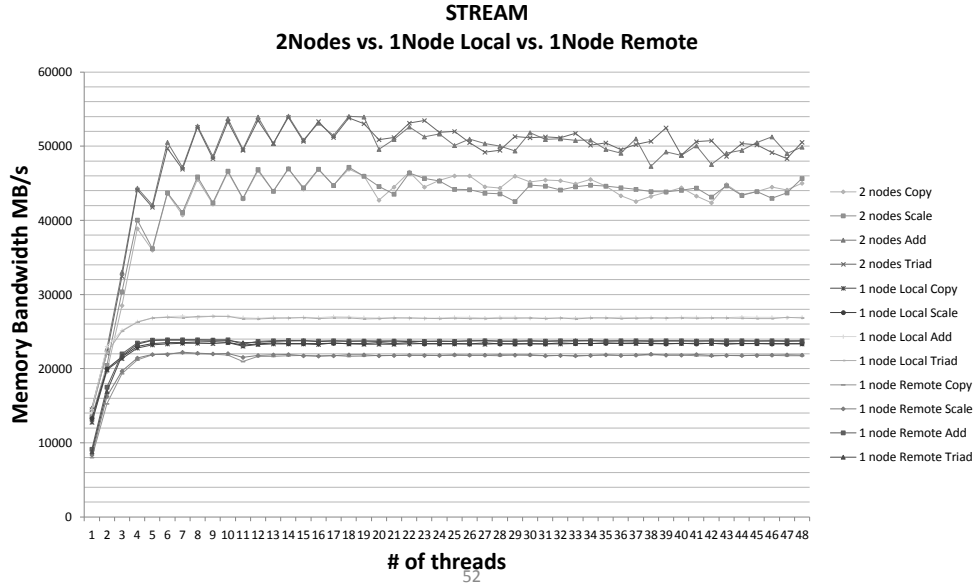
Memory Access	Memory Configuration	Activated Processor
Symmetric	2Nodes * 4x16GB DDR3 1600MHz	P0+P1
Local	1Node(#0) * 4x16GB DDR3 1600MHz	P0
Remote	1Node(#0) * 4x16GB DDR3 1600MHz	P1

	2nodes memset	2nodes memcpy	1 node local memset	1 node local memcpy	1 node remote memset	1 node remote memcpy
slow	\	\	\	\	\	\
fast	5185.7	3255.3	5147.9	3162.5	2133.4	1430.2
faster	\	\	\	\	\	\
fastest	\	\	\	\	\	\

51

Test Report – STREAM

2 nodes vs. 1 node local vs. 1 node remote

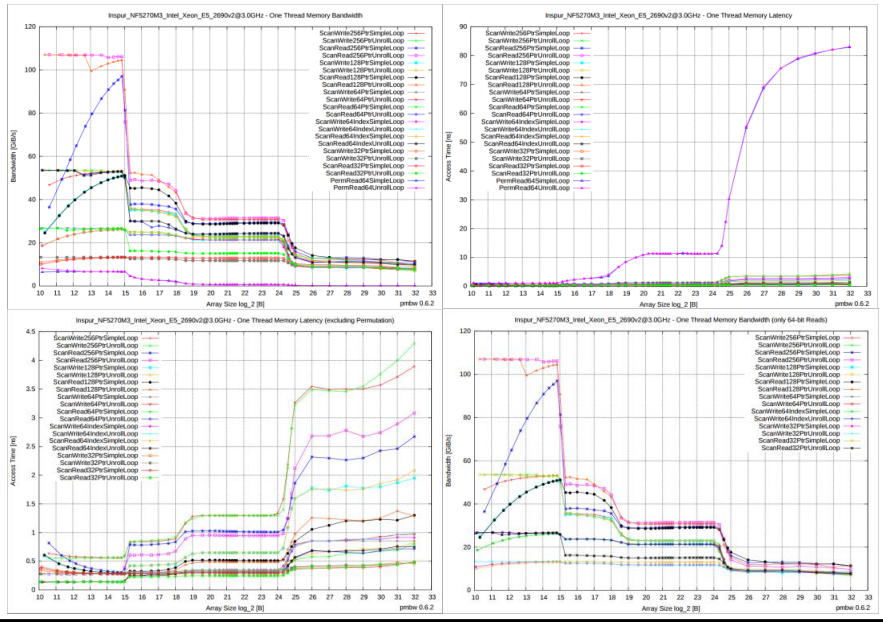


52

Test Report – pmbw(cont.)

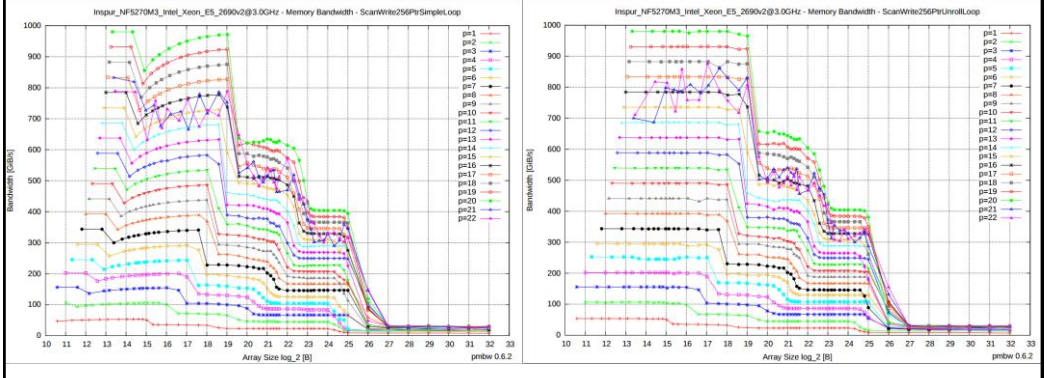
Why lines look like this?
- **Refer to Bandwidth above**

Single Thread Memory Bandwidth



Test Report – pmbw(cont.)

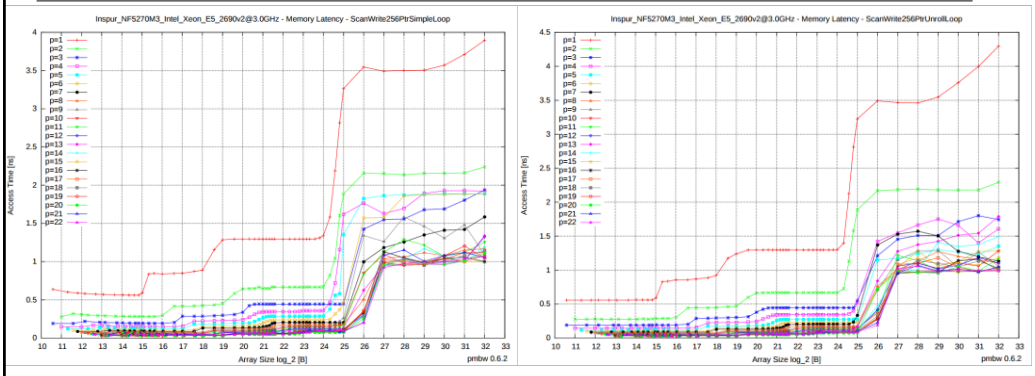
Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll



Why lines look like this?
- **P=20(parallel Cache/Memory access, 20 cores, 20 threads), so it shows best bandwidth/performance**
- **L1 = 32K, L2 = 256K, L3 = 25M(shared by all cores per Processor)**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll

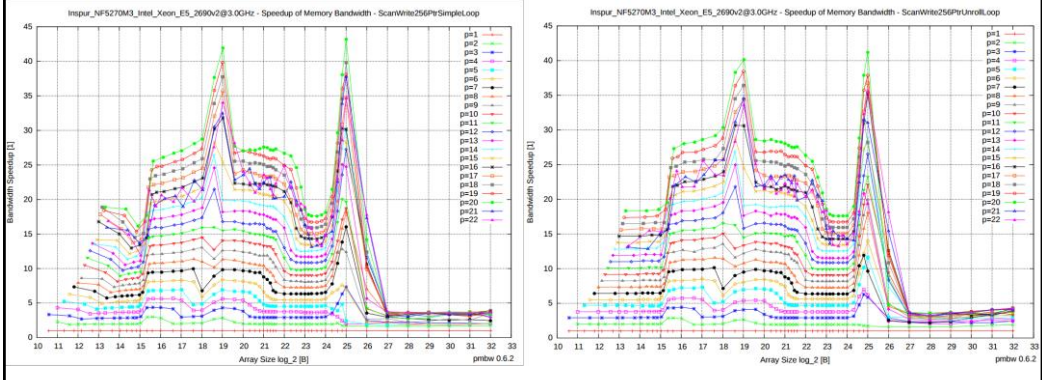


Why lines look like this?

- **Refer to Bandwidth line chart on last slides, highest bandwidth has lowest Latency accordingly**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll

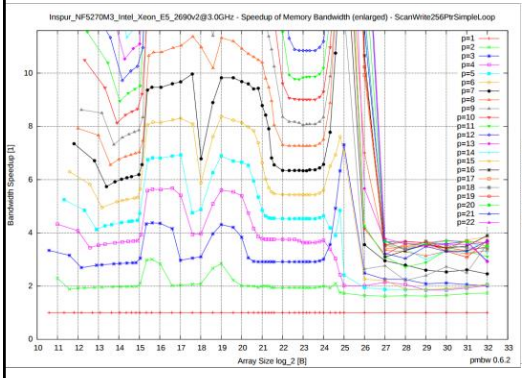


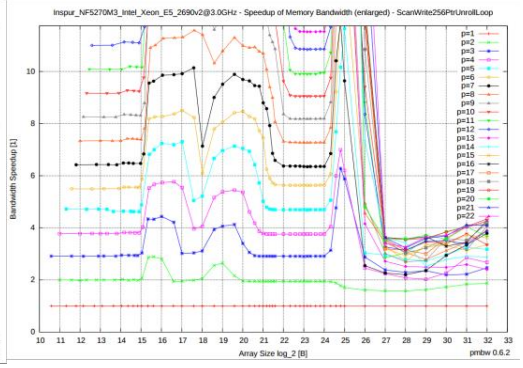
Why lines look like this?

- **Speedup bandwidth depends on Parallel Memory Access(Multiple threads, maybe, have improved ratio of cache hit)**
- **Also it is restricted to the size of L1, L2 and L3**
- **Total 20 threads(one thread per core) have best speedup bandwidth**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll





Test Report – pmbw(cont.)

Details for list below

ScanWrite256PtrSimpleLoop
ScanWrite256PtrUnrollLoop
ScanRead256PtrSimpleLoop
ScanRead256PtrUnrollLoop
ScanWrite128PtrSimpleLoop
ScanWrite128PtrUnrollLoop
ScanRead128PtrSimpleLoop
ScanRead128PtrUnrollLoop
ScanWrite64PtrSimpleLoop
ScanWrite64PtrUnrollLoop
ScanRead64PtrSimpleLoop
ScanRead64PtrUnrollLoop
ScanWrite64IndexSimpleLoop
ScanWrite64IndexUnrollLoop
ScanRead64IndexSimpleLoop
ScanRead64IndexUnrollLoop
ScanWrite32PtrSimpleLoop
ScanWrite32PtrUnrollLoop
ScanRead32PtrSimpleLoop
ScanRead32PtrUnrollLoop

plots-Inspur_NF5270M3_Intel_Xeon_E5_2690v2@3.0GHz.pdf

Test Report – lmbench

Server Info

Host		OS	Description	Processor/Memory
198	Linux	2.6.35.	old HP server + CentOS5.5	Intel Xeon E5506 @2.13GHz - 2x4cores
198	Linux	2.6.35.		
c2_novs	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch service stopped	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
c2_novsvm	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch&VM service stopped	
compu-2	Linux	3.4.103		
compu-2	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0	
compu-2	Linux	3.4.103		
dell_730	Linux	2.6.32-	Dell PowerEdge R730 + CentOS6.6	
hpdl380g9	Linux	3.10.0-	HP ProLiant DL380 Gen9 + CentOS7	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores DDR4 2133MHz
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpg9local	Linux	3.10.0-		
hpg9remot	Linux	3.10.0-		
insplocal	Linux	3.10.0-	Inspur NF5270M3 + CentOS7	Intel Xeon E5-2690 v2 @3.0GHz - 2x10cores DDR3 1600MHz
inspremot	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur.NF	Linux	2.6.32-	Inspur NF5270M3 + CentOS6.6	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
inspur.NF	Linux	2.6.32-		
vm.cp	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm.cp	Linux	3.10.0-		
vm.inspur	Linux	3.10.0	Inspur NF5270M3 + CentOS6.6 + KVM(CentOS7)	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
vm.inspur	Linux	3.10.0		
vm_dp_vm	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm_dp_vm	Linux	3.10.0-		

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Local Communication bandwidths in MB/s - bigger is better

Host	OS	Pipe	AF UNIX	TCP	File reread	Mmap reread	Bcopy (libc)	Bcopy (hand)	Mem read	Mem write
198	Linux 2.6.35.	804.	2019	209.	1526.5	2536.6	1671.9	1516.6	2573	2683.
198	Linux 2.6.35.	807.	1961	215.	1910.6	3351.4	1628.9	1533.9	2564	2676.
c2_novs	Linux 3.4.103	3894	4473	2638	4069.7	5890.2	4271.8	4318.1	8188	5611.
c2_novsvm	Linux 3.4.103	4484	4965	2974	4513.9	6568.7	5252.8	5375.5	8685	6587.
compu-2	Linux 3.4.103	1561	1891	1161	2689.9	5635.2	3340.7	2988.4	5134	3845.
compu-2	Linux 3.4.103	1585	1907	1161	2654.8	5377.2	3342.2	2992.2	5141	3853.
compu-2	Linux 3.4.103	1425	1742	1015	3203.4	8155.5	3066.8	2855.9	4789	3572.
dell_730	Linux 2.6.32-	3122	8037	2729	4842.8	7599.6	6291.0	6248.1	11.K	7880.
hpdl380g9	Linux 3.10.0-	4341	8534	5106	4905.4	5793.4	7651.4	5338.0	8529	6768.
hpdl380g9	Linux 3.10.0-	4291	8515	5347	5002.2	5819.7	7793.7	5533.8	8684	6798.
hpdl380g9	Linux 3.10.0-	4336	8638	4894	5997.2	7573.1	7588.6	5617.5	8731	6797.
hpdl380g9	Linux 3.10.0-	4302	8620	5094	6150.6	7861.3	7644.9	5412.7	8716	6810.
hpg9local	Linux 3.10.0-	4325	8405	5079	6300.7	8262.5	8057.9	5787.3	8808	7209.
hpg9remot	Linux 3.10.0-	4251	7968	5271	5345.4	6539.8	7499.9	4868.2	7191	5521.
insplocal	Linux 3.10.0-	4638	9239	5055	8000.1	12.8K	3410.2	6326.7	12.K	8062.
inspremot	Linux 3.10.0-	4726	9556	5149	4773.5	5591.4	1452.8	4070.2	6124	5101.
inspur-nf	Linux 3.10.0-	4549	9506	5209	5130.0	6336.6	3361.3	6218.8	12.K	8073.
inspur-nf	Linux 3.10.0-	4622	9510	5202	7208.6	9540.3	3363.9	6491.6	12.K	8081.
inspur-nf	Linux 3.10.0-	4586	9602	5185	5132.5	9506.2	3363.9	6392.5	12.K	8153.
inspur.NF	Linux 2.6.32-	5521	6474	1669	6983.1	12.1K	5000.9	5078.6	12.K	7288.
inspur.NF	Linux 2.6.32-	5402	6313	1670	4502.6	5537.6	5008.5	5063.6	12.K	7249.
vm.cp	Linux 3.10.0-	4105	3293	3077	5245.9	5665.9	4983.9	5180.3	7782	5138.
vm.cp	Linux 3.10.0-	5661	2308	2063	5443.1	7614.4	7732.3	4445.2	8482	6985.
vm.inspur	Linux 3.10.0	3566	3969	3356	6096.4	11.3K	5240.2	5511.9	10.K	8114.
vm.inspur	Linux 3.10.0	3449	3896	3373	6223.4	11.2K	5555.2	5465.9	10.K	8030.
vm_dp_vm	Linux 3.10.0-	4326	2171	2755	5633.8	8421.4	6328.3	5774.4	8762	7239.
vm_dp_vm	Linux 3.10.0-	5333	2131	1706	5966.7	8730.6	8187.2	5996.7	8587	7397.

Note: please ignore the error data caused by the instability of system environment !

According to the test results, focus on performance on host machine

- **Inspur NF5270M3 , there is a big gap of bandwidth between local and remote memory access**

insplocal Linux 3.10.0- 4638 9239 5055 8000.1 12.8K 3410.2 6326.7 12.K 8062.
inspremot Linux 3.10.0- 4726 9556 5149 4773.5 5591.4 1452.8 4070.2 6124 5101.

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Memory latencies in nanoseconds - smaller is better
(WARNING - may not be correct, check graphs)

Host	OS	Mhz	L1 $	L2 $	Main mem	Rand mem	Guesses
198	Linux	2.6.35.	2131	3.0850	8.1480	58.9	215.1	
198	Linux	2.6.35.	2131	3.1910	11.8	58.4	220.1	
c2_novs	Linux	3.4.103	3080	1.3200	7.3290	36.6	116.2	
c2_novsvm	Linux	3.4.103	3489	1.1700	5.8940	36.2	132.9	
compu-2	Linux	3.4.103	1188	3.4640	14.9	42.2	172.3	
compu-2	Linux	3.4.103	1188	3.4420	15.7	42.1	10.4	
compu-2	Linux	3.4.103	1189	13.9	78.8	205.6		
dell_730	Linux	2.6.32-	3094	1.2930	4.6690	22.9	79.5	
hpdl380g9	Linux	3.10.0-	3493	1.1460	5.4760	33.2		
hpdl380g9	Linux	3.10.0-	3493	1.1470	5.7150	35.0	115.1	
hpdl380g9	Linux	3.10.0-	3493	1.1460	5.4980	35.1	113.8	
hpdl380g9	Linux	3.10.0-	3493	1.1470	5.7040	33.2	114.5	
hpg9local	Linux	3.10.0-	3493	1.1460	4.6780	31.7	110.5	
hpg9remot	Linux	3.10.0-	3493	1.1450	5.6430	41.0	165.8	
insplocal	Linux	3.10.0-	3602	1.1130	4.5860	18.8	81.3	
inspremot	Linux	3.10.0-	3602	1.1130	3.3400	50.9	195.1	
inspur-nf	Linux	3.10.0-	3602	1.1130	3.3390	18.7	80.6	
inspur-nf	Linux	3.10.0-	3602	1.1120	3.3420	18.7	81.0	
inspur-nf	Linux	3.10.0-	3602	1.1120	3.3410	18.5	81.9	
inspur.NF	Linux	2.6.32-	3012	1.2130	5.2230	19.3	84.5	
inspur.NF	Linux	2.6.32-	3012	1.2130	6.8120	19.3	84.5	
vm.cp	Linux	3.10.0-	3079	1.5880	11.6	65.4	143.6	
vm.cp	Linux	3.10.0-	3079	1.4700	5.7840	30.4	104.4	
vm.inspur	Linux	3.10.0	2885	1.3890	4.3220	22.5	90.9	
vm.inspur	Linux	3.10.0	2885	1.3910	4.2290	21.0	90.8	
vm_dp_vm	Linux	3.10.0-	3279	1.2260	6.4110	30.4	97.8	
vm_dp_vm	Linux	3.10.0-	3279	1.1520	8.8750	29.1	96.8	

For memoey latencies,

- **Inspur NF5270M3 has the best performance on both 2 nodes and local memory access, but has worst performance on remote memory access. (It seems there is the PCB design issue?)**

61

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Micro-benchmarking Details

results.txt

62

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3

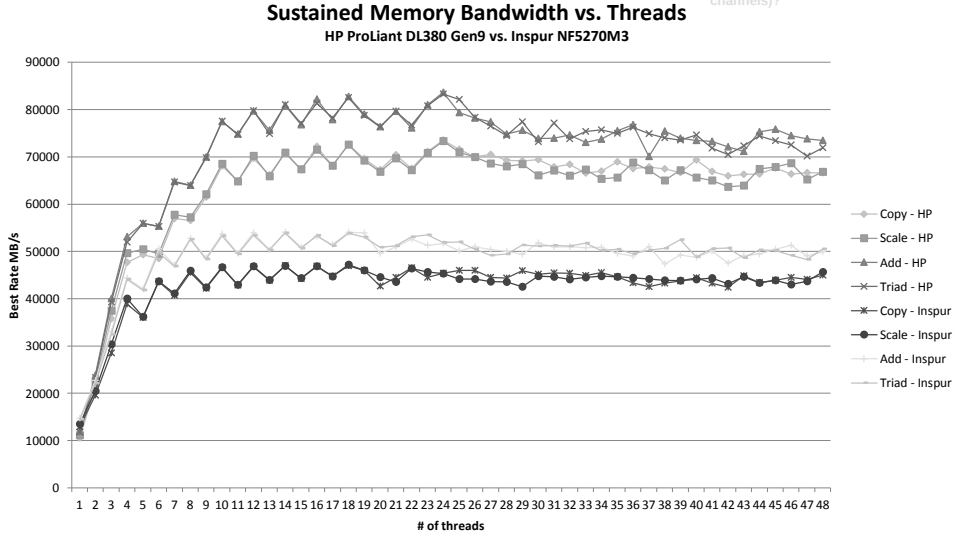
Test Report – STREAM

#define STREAM_TYPE double

Why lines look like this?

HP vs. Inspur on Best Rate. Factors,

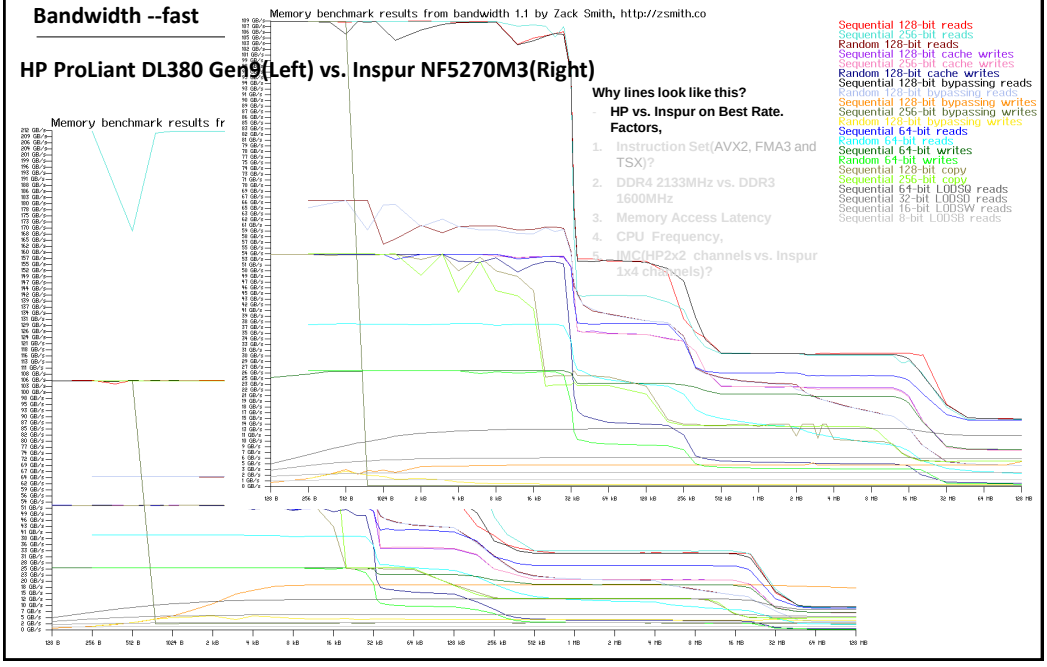
1. DDR4 2133MHz vs. DDR3 1600MHz
2. Memory Access Latency
3. CPU Frequency.
4. IMC(HP2x2 channels vs. Inspur 1x4 channels)?



Test Report – Bandwidth

Bandwidth --fast

HP ProLiant DL380 Gen9(Left) vs. Inspur NF5270M3(Right)



Why lines look like this?

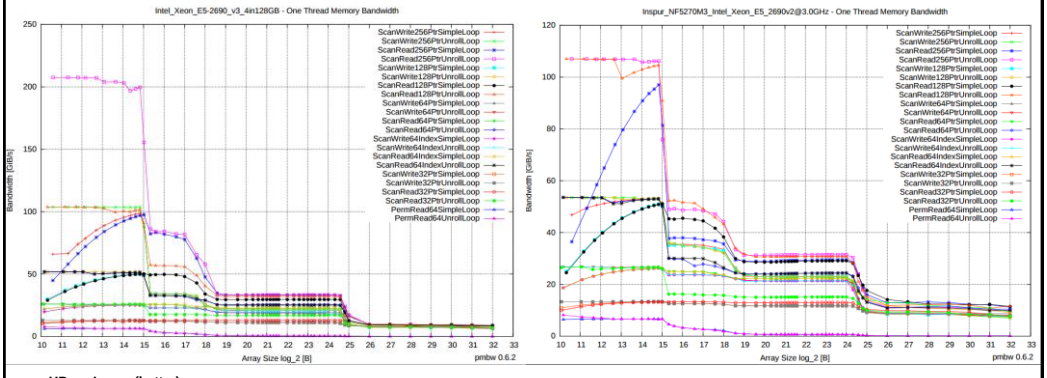
- **HP vs. Inspur on Best Rate. Factors,**
  1. Instruction Set(AVX2, FMA3 and TSX)?
  2. DDR4 2133MHz vs. DDR3 1600MHz
  3. Memory Access Latency
  4. CPU Frequency,
  5. IMC(HP2x2 channels vs. Inspur 1x4 channels)?

Test Report – pmbw

HP vs.Inspur - Single Thread Memory Bandwidth

Why lines look like this?

- **Refer to Bandwidth above**



HP vs. Inspur(better)

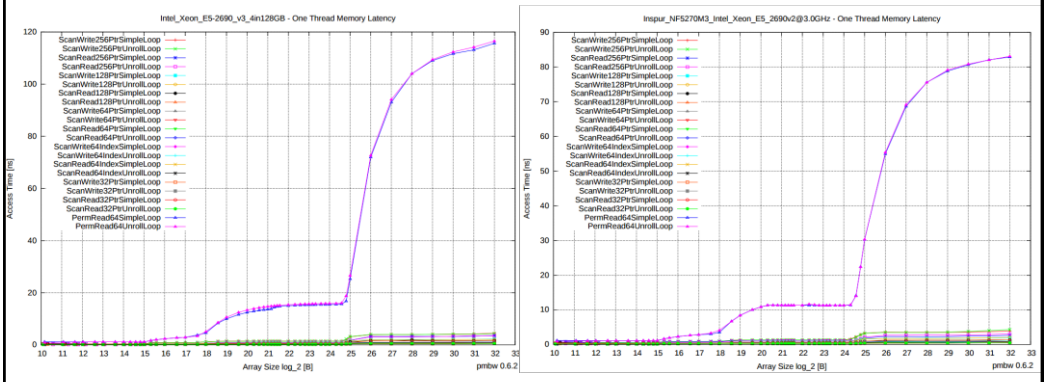
BW of L1 cache: HP

- Instruction Set(AVX2, FMA3 and TSX)?

Test Report – pmbw(cont.)

Why lines look like this?
- **Refer to Bandwidth above**

HP vs.Inspur - Single Thread Memory Bandwidth



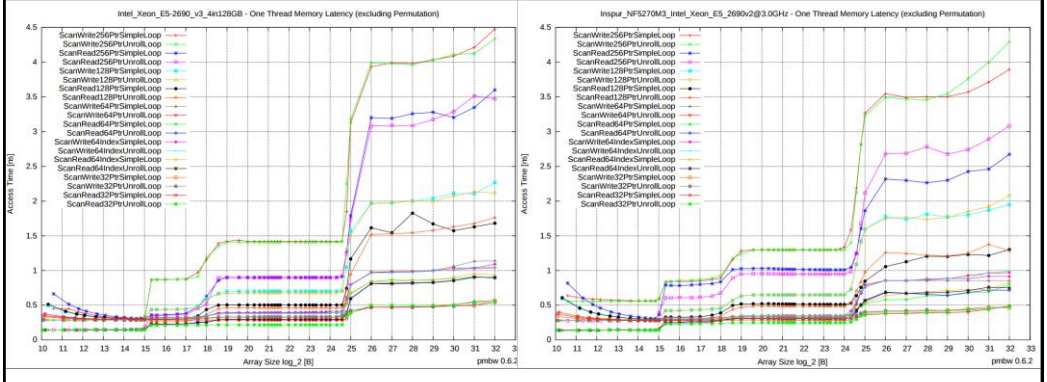
HP vs. Inspur(better)

Access Time: Inspur
- CPU Frequency 3.0GHz
- 1x4 channels Integrated Memory Controller?

Test Report – pmbw(cont.)

Why lines look like this?
- **Refer to Bandwidth above**

HP vs.Inspur - Single Thread Memory Bandwidth



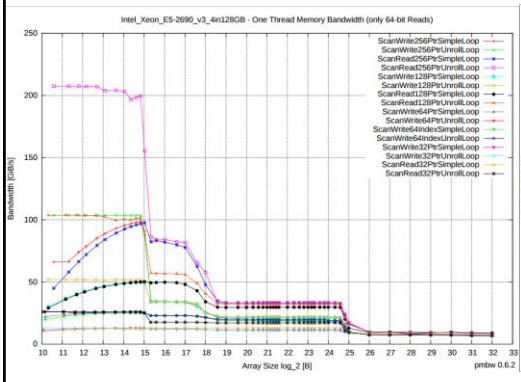
HP vs. Inspur(better)

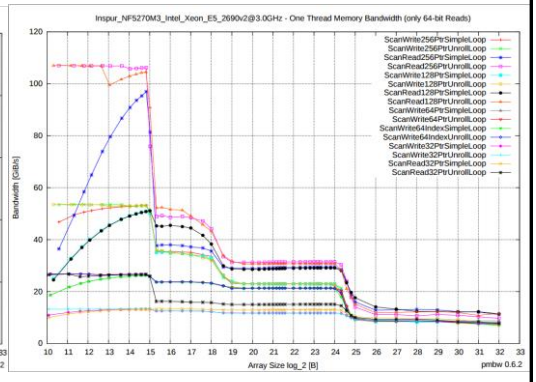
Access Time: Inspur
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

Test Report – pmbw(cont.)

Why lines look like this?
Refer to Bandwidth above

HP vs.Inspur - Single Thread Memory Bandwidth





HP vs. Inspur(better)

BW of L1 cache: HP

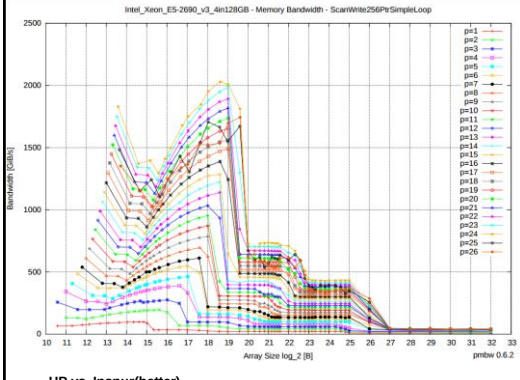
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- channels of IMC?

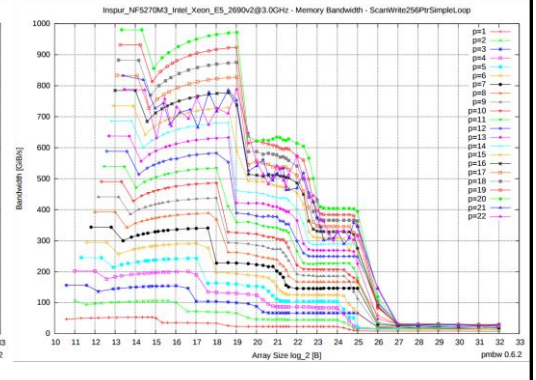
69

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

BW of L1 cache: HP

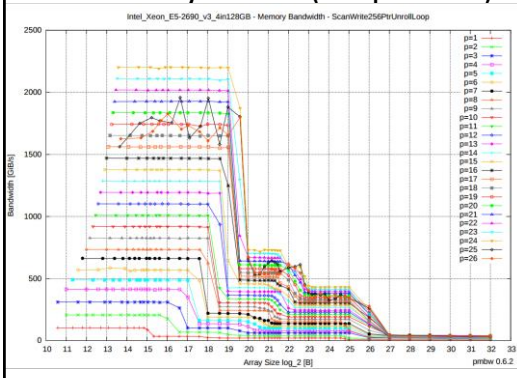
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

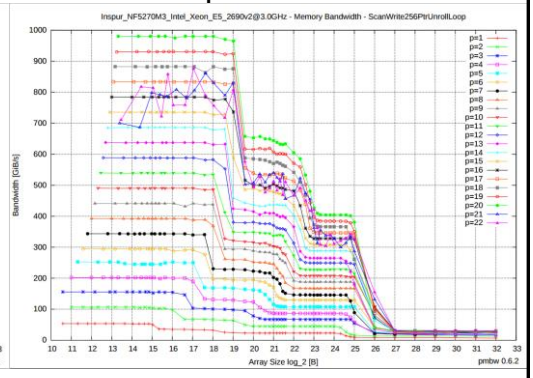
70

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

BW of L1 cache: HP

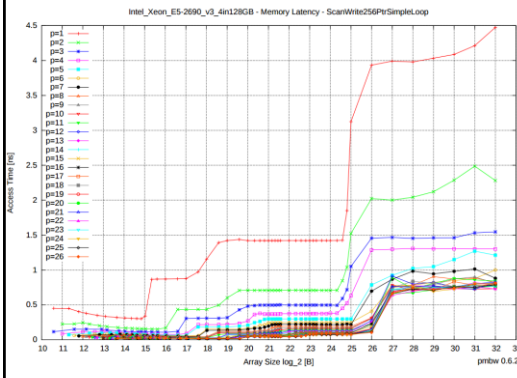
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

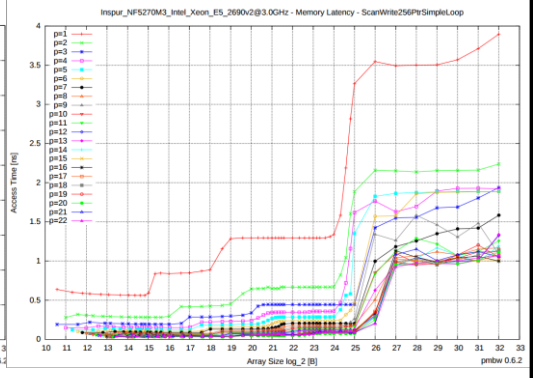
71

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

Access Time: Inspur

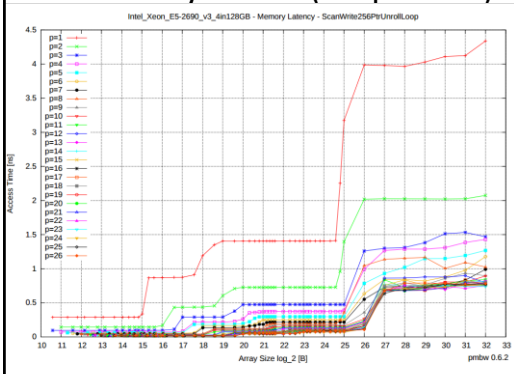
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

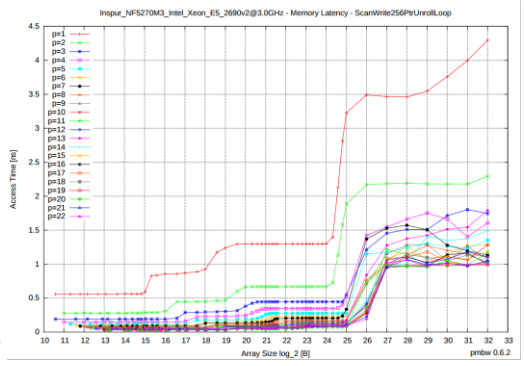
72

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

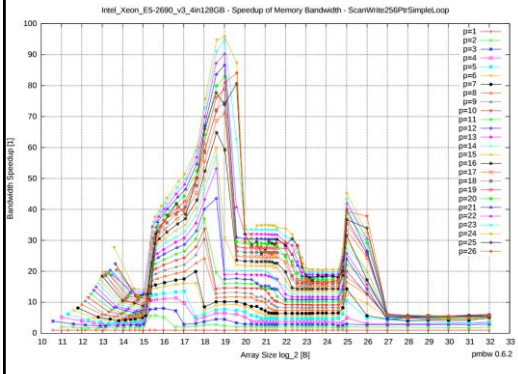
Access Time:

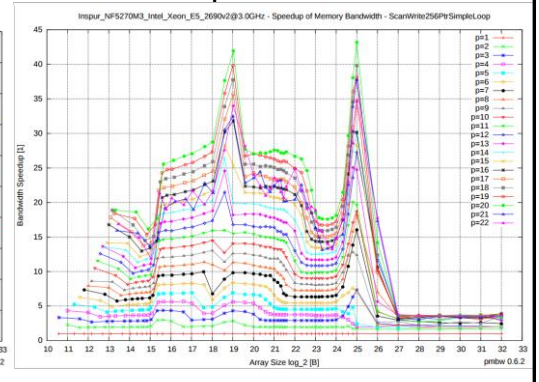
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

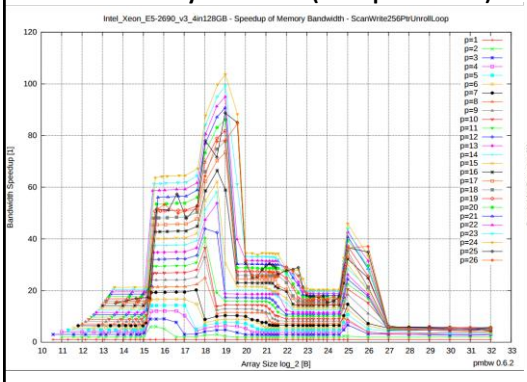
BW of L1 cache: HP

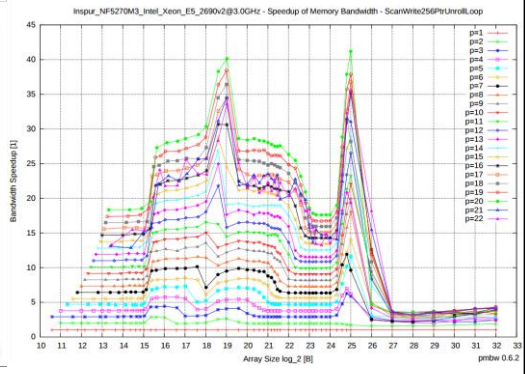
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll



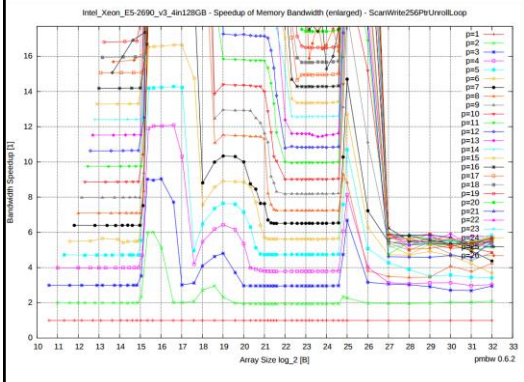


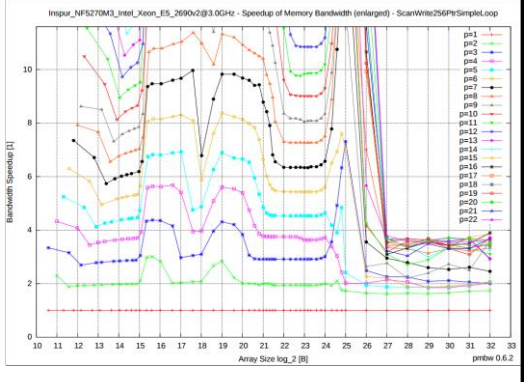
75

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll



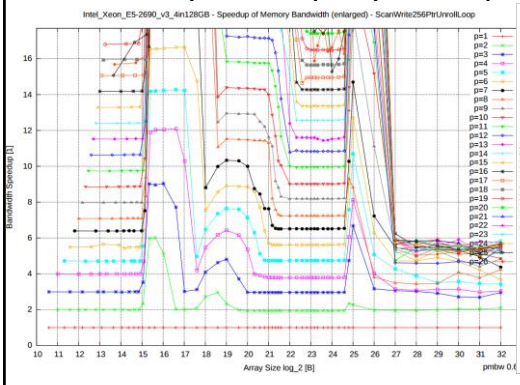


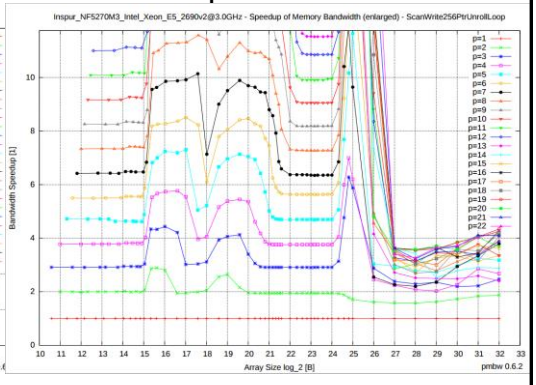
76

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





Test Report – lmbench

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3

```
*Local* Communication bandwidths in MB/s - bigger is better
-----------------------------------------------------------------------------
Host                OS  Pipe AF    TCP  File   Mmap  Bcopy  Bcopy  Mem   Mem
                             UNIX      reread reread (libc) (hand) read write
--------- ------------- ---- ---- ---- ------ ------ ------ ------ ---- -----
hpdl380g9 Linux 3.10.0- 4291 8515 5347 5002.2 5819.7 7793.7 5533.8 8684 6798.
hpdl380g9 Linux 3.10.0- 4336 8638 4894 5997.2 7573.1 7588.6 5617.5 8731 6797.
hpdl380g9 Linux 3.10.0- 4302 8620 5094 6150.6 7861.3 7644.9 5412.7 8716 6810.
hpg9local Linux 3.10.0- 4325 8405 5079 6300.7 8262.5 8057.9 5787.3 8808 7209.
hpg9remot Linux 3.10.0- 4251 7968 5271 5345.4 6539.8 7499.9 4868.2 7191 5521.
insplocal Linux 3.10.0- 4638 9239 5055 8000.1  12.8K 3410.2 6326.7 12.K 8062.
inspremot Linux 3.10.0- 4726 9556 5149 4773.5 5591.4 1452.8 4070.2 6124 5101.
inspur-nf Linux 3.10.0- 4549 9506 5209 5130.0 6336.6 3361.3 6218.8 12.K 8073.
inspur-nf Linux 3.10.0- 4622 9510 5202 7208.6 9540.3 3363.9 6491.6 12.K 8081.
inspur-nf Linux 3.10.0- 4586 9602 5185 5132.5 9506.2 3363.9 6392.5 12.K 8153.

Memory latencies in nanoseconds - smaller is better
    (WARNING - may not be correct, check graphs)
-----------------------------------------------------------------------------
Host                 OS   Mhz   L1 $   L2 $    Main mem    Rand mem    Guesses
--------- ------------- ----   ----   ----    --------    --------    -------
hpdl380g9 Linux 3.10.0-  3493 1.1470 5.7150   35.0        115.1
hpdl380g9 Linux 3.10.0-  3493 1.1460 5.4980   35.1        113.8
hpdl380g9 Linux 3.10.0-  3493 1.1470 5.7040   33.2        114.5
hpg9local Linux 3.10.0-  3493 1.1460 4.6780   31.7        110.5
hpg9remot Linux 3.10.0-  3493 1.1450 5.6430   41.0        165.8
insplocal Linux 3.10.0-  3602 1.1130 4.5860   18.8         81.3
inspremot Linux 3.10.0-  3602 1.1130 3.3400   50.9        195.1
inspur-nf Linux 3.10.0-  3602 1.1130 3.3390   18.7         80.6
inspur-nf Linux 3.10.0-  3602 1.1120 3.3420   18.7         81.0
inspur-nf Linux 3.10.0-  3602 1.1120 3.3410   18.5         81.9
```

results.txt

Communication bandwidth

- **HP is better on Mmap and Bcopy**
- **Inspur is better on Mem read and Mem write**
- **For remote memory access, HP is better.**

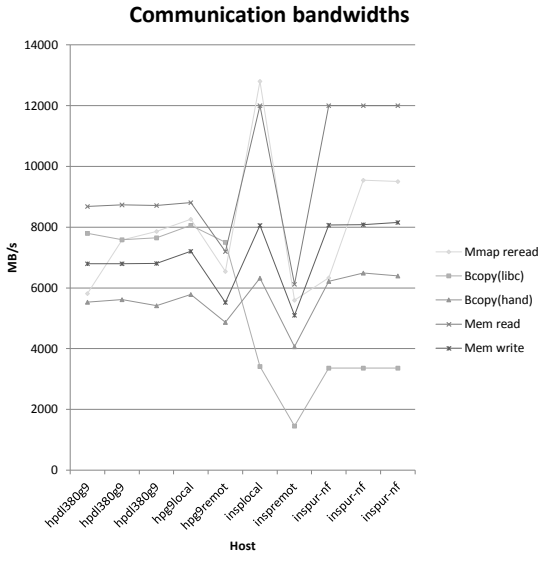
Memoey latencies

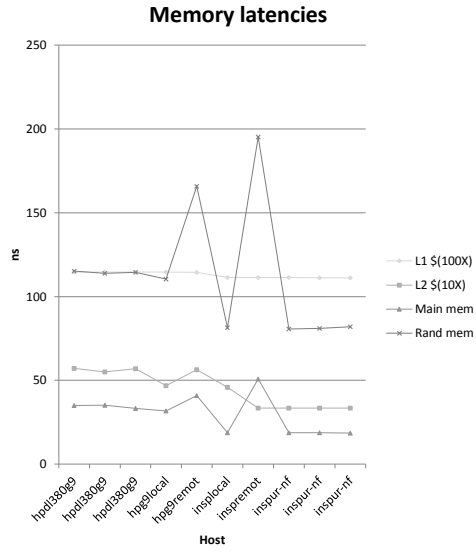
- **HP has a bit of latencies more than Inspur on L1 and L2 cache, more latencies on memory subsequential and random memory access**
- **For remote access memory latencies, HP is also better.**

Host		OS	Description	Processor	Memory
hpdl380g9	Linux	3.10.0-			
hpdl380g9	Linux	3.10.0-			
hpdl380g9	Linux	3.10.0-	HP ProLiant	Intel Xeon E5-2690	DDR4 2133MHz
hpdl380g9	Linux	3.10.0-	DL380 Gen9 +	v3 @2.6GHz -	2x4x16GB
hpg9local	Linux	3.10.0-	CentOS7	2x12cores	
hpg9remot	Linux	3.10.0-			
insplocal	Linux	3.10.0-			
inspremot	Linux	3.10.0-	Inspur	Intel Xeon E5-2690	DDR3 1600MHz
inspur-nf	Linux	3.10.0-	NF5270M3 +	v2 @3.0GHz -	2x4x16GB
inspur-nf	Linux	3.10.0-	CentOS7	2x10cores	
inspur-nf	Linux	3.10.0-			

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3





79

Sugon I620-G20

80

System Info

Inspur NF5270M3

- CPU (http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon%20E5-2670%20v3.html)
  - 2 x Intel(R) Xeon(R) CPU E5-2670 v3@ 2.30GHz
  - 2 x 12cores, 64bit(CPU0 presented)
- Bus Speed
  - 9.6 GT/s QPI (4800 MHz)
  - 5 GT/s DMI
- Memory
  - L1:
    12 x 32 KB 8-way set associative instruction caches
    12 x 32 KB 8-way set associative data caches
  - L2:
    12 x 256 KB 8-way set associative caches
  - L3:
    30 MB 20-way set associative shared cache
  - 4 x 16GB 2Rx4 PC4-2133P RAM(DDR4 2133MHz)

```
[root@localhost results]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11
node 0 size: 65421 MB
node 0 free: 60308 MB
node distances:
node   0
  0:  10
[root@localhost results]#
```

- OS: CentOS7
  - [root@localhost STREAM]# uname -o
  - GNU/Linux
  - [root@localhost STREAM]# uname -r
  - 3.10.0-229.el7.x86_64
  - root@localhost STREAM]# gcc –v
  - …
  - gcc version 4.8.3 20140911 (Red Hat 4.8.3-9) (GCC)

Pre-configuration

- 2 Processors to set scaling_governor == performance [i=0~19]

  echo performance > /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor

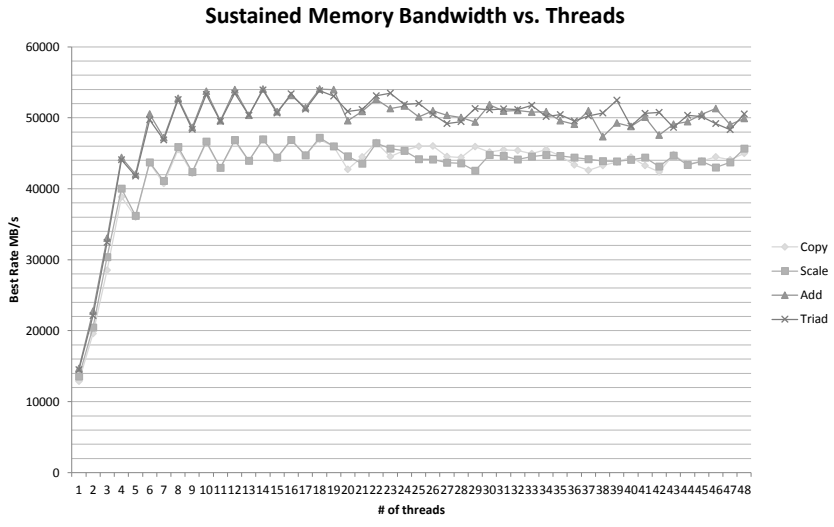
- Optimized memory config recommended (1 nodes: 4x16GB on each DIMM0 of ch1, ch2, ch3 and ch4 per processor)

Test Report – STREAM

#define STREAM_TYPE double

Why lines look like this?

- **# of Memory channel == 4 (bandwidth grows linearly until reached channel number restriction, e.g. From 10000 to 50000 MB/s, threads == 4)**
- **4 x16GB Memory/channel/Processor**
- **2 x 12 cores(best performance when up to 24 threads)**



Test Report – STREAM（cont.）

- STREAM log

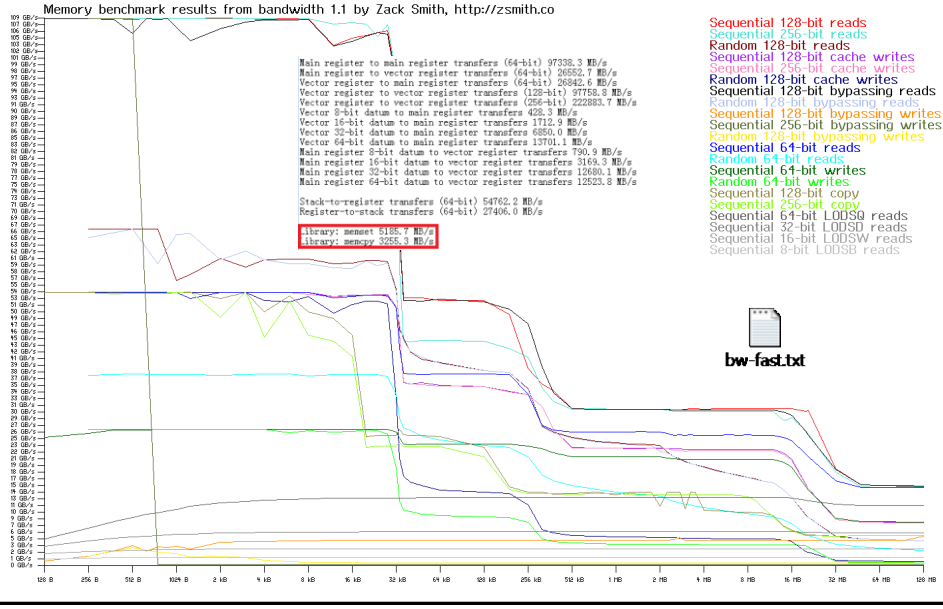
Test Report – Bandwidth

Bandwidth --slow (high precision, ~8hrs)

- Not Tested

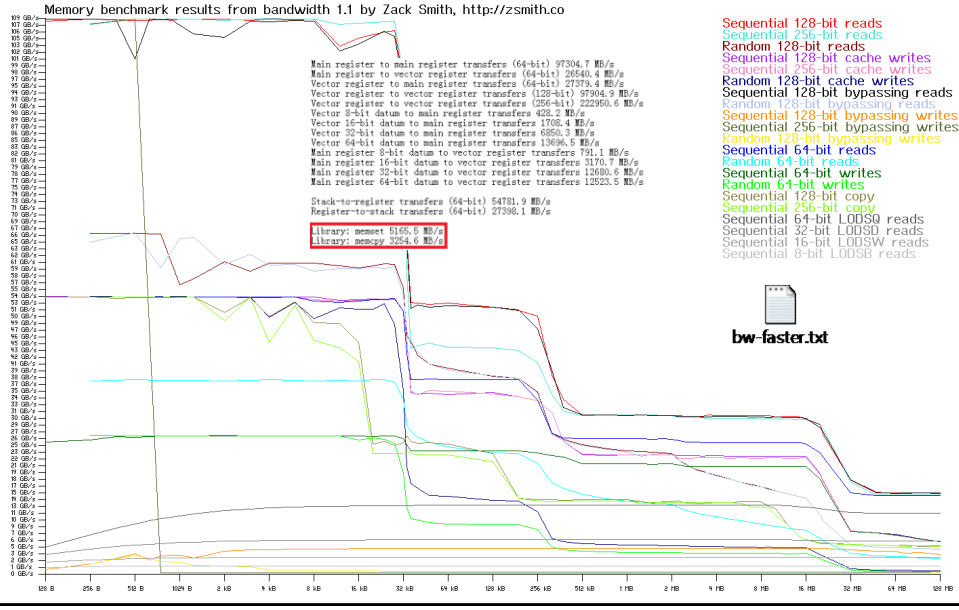
Test Report – Bandwidth(cont.)

Bandwidth --fast



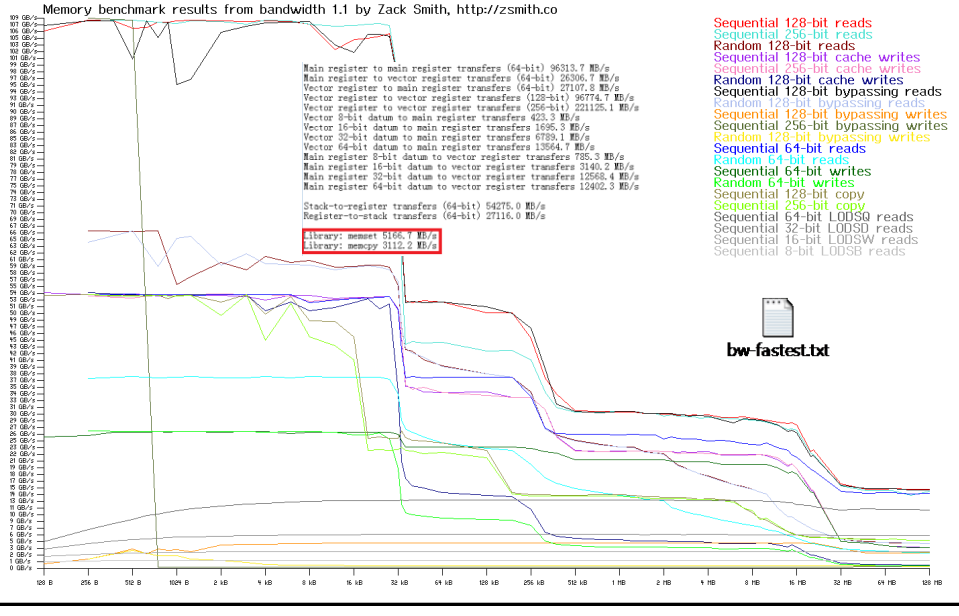
Test Report – Bandwidth(cont.)

Bandwidth --faster



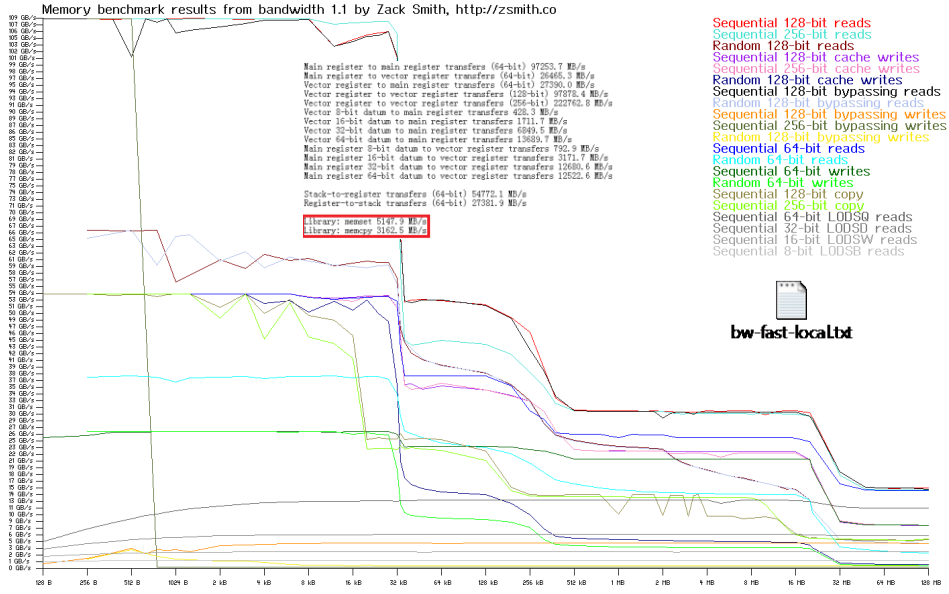
Test Report – Bandwidth(cont.)

Bandwidth --fastest



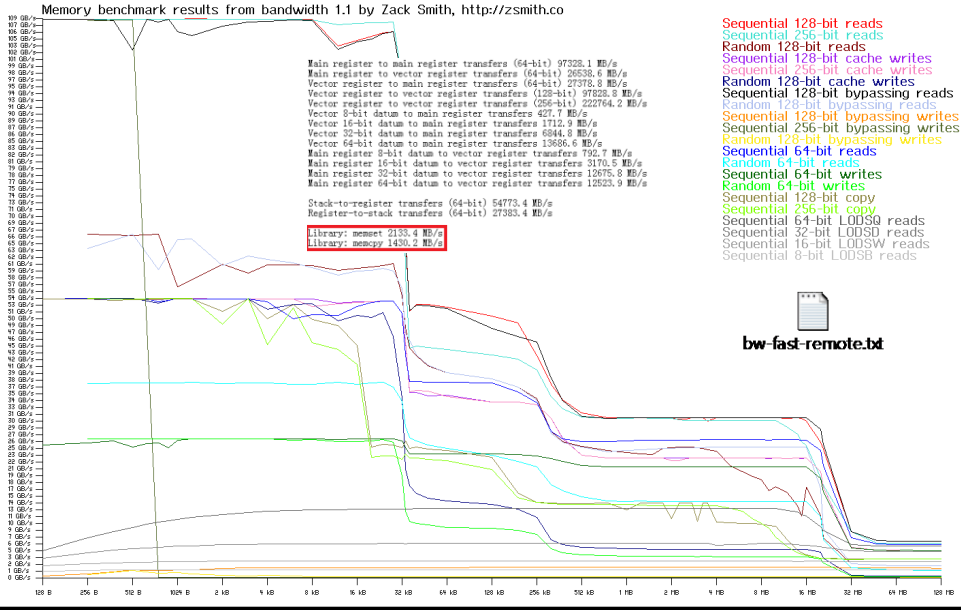
Test Report – Local Memory Access

Bandwidth –fast



Test Report – Remote Memory Access via QPI

Bandwidth --fast



Test Report – Bandwidth

2Nodes vs 1Node Remote/Local

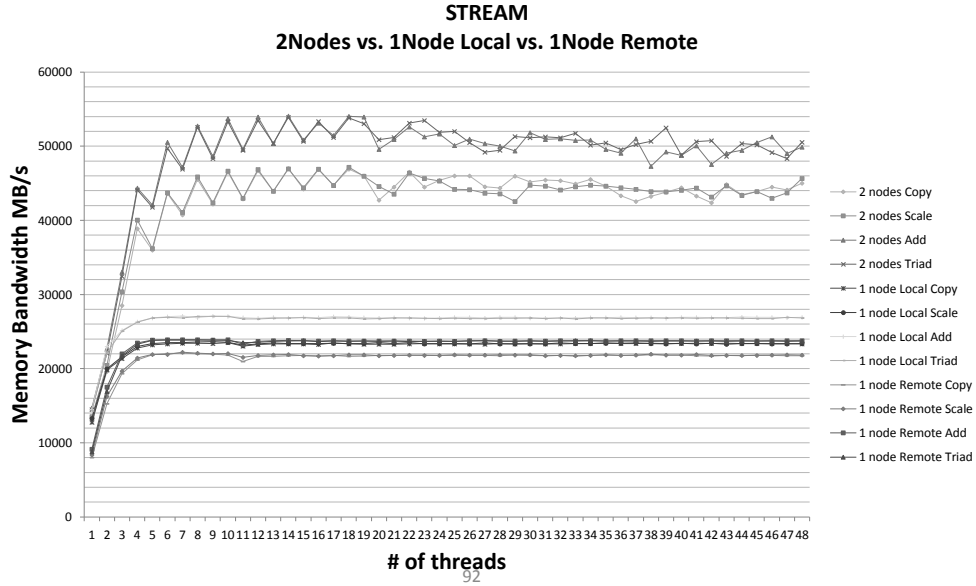
Memory Access	Memory Configuration	Activated Processor
Symmetric	2Nodes * 4x16GB DDR3 1600MHz	P0+P1
Local	1Node(#0) * 4x16GB DDR3 1600MHz	P0
Remote	1Node(#0) * 4x16GB DDR3 1600MHz	P1

	2nodes memset	2nodes memcpy	1 node local memset	1 node local memcpy	1 node remote memset	1 node remote memcpy
slow	\	\	\	\	\	\
fast	5185.7	3255.3	5147.9	3162.5	2133.4	1430.2
faster	\	\	\	\	\	\
fastest	\	\	\	\	\	\

91

Test Report – STREAM

2 nodes vs. 1 node local vs. 1 node remote

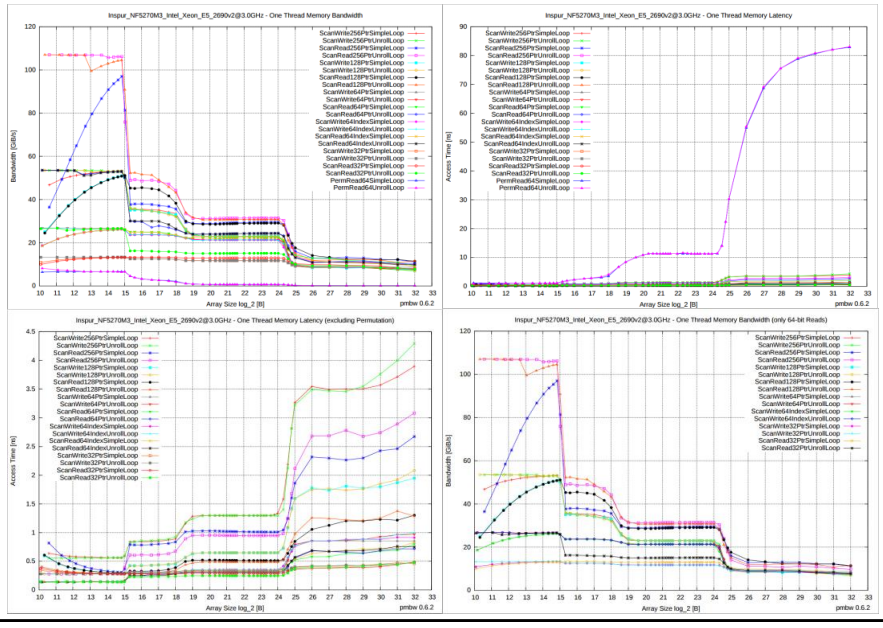


92

Test Report – pmbw(cont.)

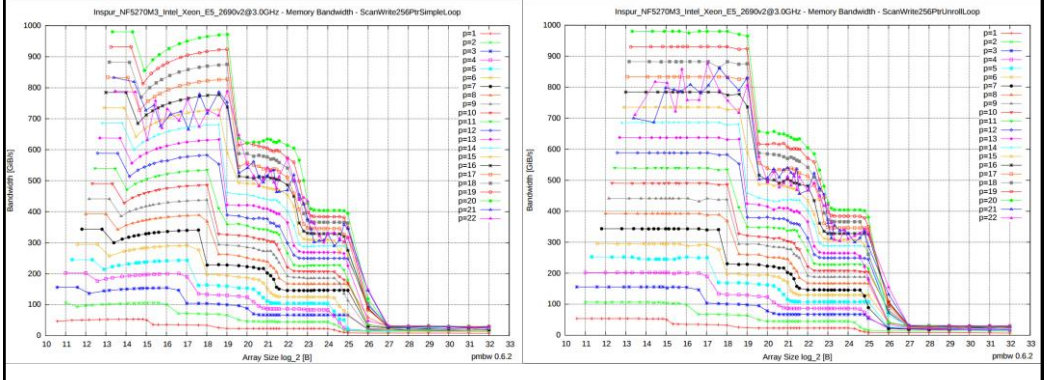
Single Thread Memory Bandwidth

Why lines look like this?
- Refer to Bandwidth above



Test Report – pmbw(cont.)

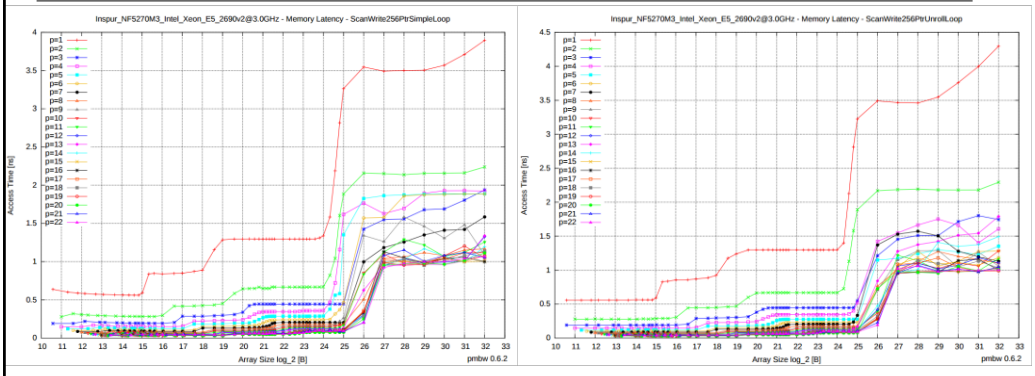
Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll



Why lines look like this?
- P=20(parallel Cache/Memory access, 20 cores, 20 threads), so it shows best bandwidth/performance
- L1 = 32K, L2 = 256K, L3 = 25M(shared by all cores per Processor)

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll

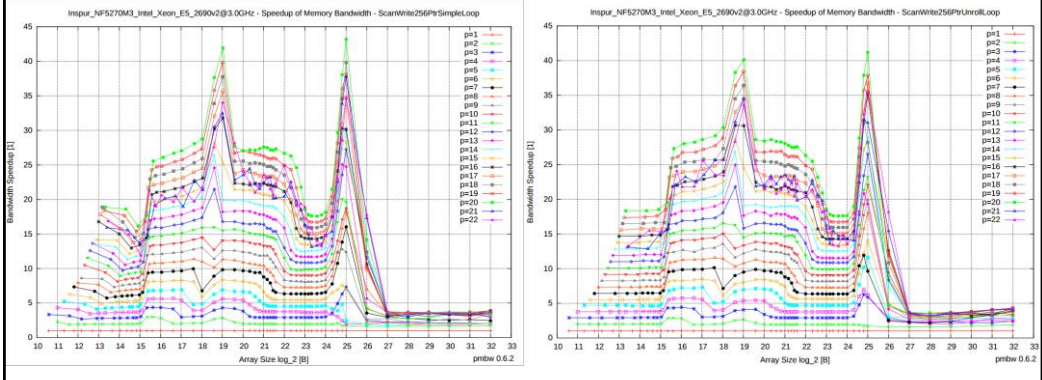


Why lines look like this?

- **Refer to Bandwidth line chart on last slides, highest bandwidth has lowest Latency accordingly**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll

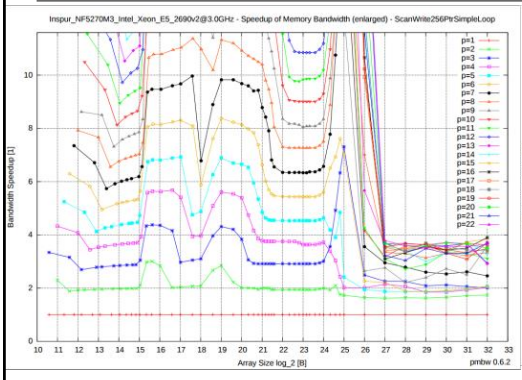


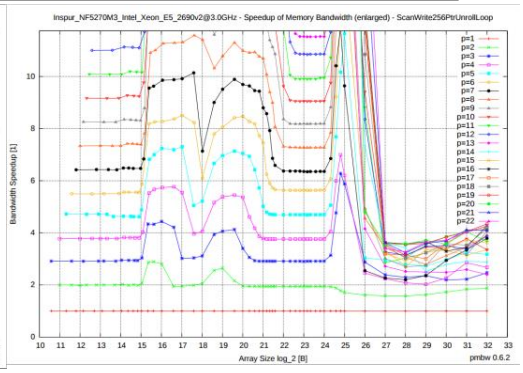
Why lines look like this?

- **Speedup bandwidth depends on Parallel Memory Access(Multiple threads, maybe, have improved ratio of cache hit)**
- **Also it is restricted to the size of L1, L2 and L3**
- **Total 20 threads(one thread per core) have best speedup bandwidth**

Test Report – pmbw(cont.)

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr Simple vs. Unroll





Test Report – pmbw(cont.)

Details for list below

ScanWrite256PtrSimpleLoop
ScanWrite256PtrUnrollLoop
ScanRead256PtrSimpleLoop
ScanRead256PtrUnrollLoop
ScanWrite128PtrSimpleLoop
ScanWrite128PtrUnrollLoop
ScanRead128PtrSimpleLoop
ScanRead128PtrUnrollLoop
ScanWrite64PtrSimpleLoop
ScanWrite64PtrUnrollLoop
ScanRead64PtrSimpleLoop
ScanRead64PtrUnrollLoop
ScanWrite64IndexSimpleLoop
ScanWrite64IndexUnrollLoop
ScanRead64IndexSimpleLoop
ScanRead64IndexUnrollLoop
ScanWrite32PtrSimpleLoop
ScanWrite32PtrUnrollLoop
ScanRead32PtrSimpleLoop
ScanRead32PtrUnrollLoop

plots-Inspur_NF5270M3_Intel_Xeon_E5_2690v2@3.0GHz.pdf

Test Report – lmbench

Server Info

Host		OS	Description	Processor/Memory
198	Linux	2.6.35.	old HP server + CentOS5.5	Intel Xeon E5506 @2.13GHz - 2x4cores
198	Linux	2.6.35.		
c2_novs	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch service stopped	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
c2_novsvm	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + vswitch&VM service stopped	
compu-2	Linux	3.4.103		
compu-2	Linux	3.4.103	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0	
compu-2	Linux	3.4.103		
dell_730	Linux	2.6.32-	Dell PowerEdge R730 + CentOS6.6	
hpdl380g9	Linux	3.10.0-	HP ProLiant DL380 Gen9 + CentOS7	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores DDR4 2133MHz
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpdl380g9	Linux	3.10.0-		
hpg9local	Linux	3.10.0-		
hpg9remot	Linux	3.10.0-		
insplocal	Linux	3.10.0-	Inspur NF5270M3 + CentOS7	Intel Xeon E5-2690 v2 @3.0GHz - 2x10cores DDR3 1600MHz
inspremot	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur-nf	Linux	3.10.0-		
inspur.NF	Linux	2.6.32-	Inspur NF5270M3 + CentOS6.6	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
inspur.NF	Linux	2.6.32-		
vm.cp	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm.cp	Linux	3.10.0-		
vm.inspur	Linux	3.10.0	Inspur NF5270M3 + CentOS6.6 + KVM(CentOS7)	Intel Xeon E5-2670 v2 @2.5GHz - 2x10cores
vm.inspur	Linux	3.10.0		
vm_dp_vm	Linux	3.10.0-	HP ProLiant DL380 Gen9 + Wind River Linux 5.0.1.0 + VM	Intel Xeon E5-2690 v3 @2.6GHz - 2x12cores
vm_dp_vm	Linux	3.10.0-		

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Local Communication bandwidths in MB/s - bigger is better

Host	OS	Pipe	AF UNIX	TCP	File reread	Mmap reread	Bcopy (libc)	Bcopy (hand)	Mem read	Mem write
198	Linux 2.6.35.	804.	2019	209.	1526.5	2536.6	1671.9	1516.6	2573	2683.
198	Linux 2.6.35.	807.	1961	215.	1910.6	3351.4	1628.9	1533.9	2564	2676.
c2_novs	Linux 3.4.103	3894	4473	2638	4069.7	5890.2	4271.8	4318.1	8188	5611.
c2_novsvm	Linux 3.4.103	4484	4965	2974	4513.9	6568.7	5252.8	5375.5	8685	6587.
compu-2	Linux 3.4.103	1561	1891	1161	2689.9	5635.2	3340.7	2988.4	5134	3845.
compu-2	Linux 3.4.103	1585	1907	1161	2654.8	5377.2	3342.2	2992.2	5141	3853.
compu-2	Linux 3.4.103	1425	1742	1015	3203.4	8155.5	3066.8	2855.9	4789	3572.
dell_730	Linux 2.6.32-	3122	8037	2729	4842.8	7599.6	6291.0	6248.1	11.K	7880.
hpdl380g9	Linux 3.10.0-	4341	8534	5106	4905.4	5793.4	7651.4	5338.0	8529	6768.
hpdl380g9	Linux 3.10.0-	4291	8515	5347	5002.2	5819.7	7793.7	5533.8	8684	6798.
hpdl380g9	Linux 3.10.0-	4336	8638	4894	5997.2	7573.1	7588.6	5617.5	8731	6797.
hpdl380g9	Linux 3.10.0-	4302	8620	5094	6150.6	7861.3	7644.9	5412.7	8716	6810.
hpg9local	Linux 3.10.0-	4325	8405	5079	6300.7	8262.5	8057.9	5787.3	8808	7209.
hpg9remot	Linux 3.10.0-	4251	7968	5271	5345.4	6539.8	7499.9	4868.2	7191	5521.
insplocal	Linux 3.10.0-	4638	9239	5055	8000.1	12.8K	3410.2	6326.7	12.K	8062.
inspremot	Linux 3.10.0-	4726	9556	5149	4773.5	5591.4	1452.8	4070.2	6124	5101.
inspur-nf	Linux 3.10.0-	4549	9506	5209	5130.0	6336.6	3361.3	6218.8	12.K	8073.
inspur-nf	Linux 3.10.0-	4622	9510	5202	7208.6	9540.3	3363.9	6491.6	12.K	8081.
inspur-nf	Linux 3.10.0-	4586	9602	5185	5132.5	9506.2	3363.9	6392.5	12.K	8153.
inspur.NF	Linux 2.6.32-	5521	6474	1669	6983.1	12.1K	5000.9	5078.6	12.K	7288.
inspur.NF	Linux 2.6.32-	5402	6313	1670	4502.6	5537.6	5008.5	5063.6	12.K	7249.
vm.cp	Linux 3.10.0-	4105	3293	3077	5245.9	5665.9	4983.9	5180.3	7782	5138.
vm.cp	Linux 3.10.0-	5661	2308	2063	5443.1	7614.4	7732.3	4445.2	8482	6985.
vm.inspur	Linux 3.10.0	3566	3969	3356	6096.4	11.3K	5240.2	5511.9	10.K	8114.
vm.inspur	Linux 3.10.0	3449	3896	3373	6223.4	11.2K	5555.2	5465.9	10.K	8030.
vm_dp_vm	Linux 3.10.0-	4326	2171	2755	5633.8	8421.4	6328.3	5774.4	8762	7239.
vm_dp_vm	Linux 3.10.0-	5333	2131	1706	5966.7	8730.6	8187.2	5996.7	8587	7397.

Note: please ignore the error data caused by the instability of system environment !

According to the test results, focus on performance on host machine

- **Inspur NF5270M3 , there is a big gap of bandwidth between local and remote memory access**

insplocal Linux 3.10.0- 4638 9239 5055 8000.1 12.8K 3410.2 6326.7 12.K 8062.
inspremot Linux 3.10.0- 4726 9556 5149 4773.5 5591.4 1452.8 4070.2 6124 5101.

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Memory latencies in nanoseconds - smaller is better
(WARNING - may not be correct, check graphs)

Host	OS	Mhz	L1 $	L2 $	Main mem	Rand mem	Guesses
198	Linux	2.6.35.	2131	3.0850	8.1480	58.9	215.1	
198	Linux	2.6.35.	2131	3.1910	11.8	58.4	220.1	
c2_novs	Linux	3.4.103	3080	1.3200	7.3290	36.6	116.2	
c2_novsvm	Linux	3.4.103	3489	1.1700	5.8940	36.2	132.9	
compu-2	Linux	3.4.103	1188	3.4640	14.9	42.2	172.3	
compu-2	Linux	3.4.103	1188	3.4420	15.7	42.1	10.4	
compu-2	Linux	3.4.103	1189	13.9	78.8	205.6		
dell_730	Linux	2.6.32-	3094	1.2930	4.6690	22.9	79.5	
hpdl380g9	Linux	3.10.0-	3493	1.1460	5.4760	33.2		
hpdl380g9	Linux	3.10.0-	3493	1.1470	5.7150	35.0	115.1	
hpdl380g9	Linux	3.10.0-	3493	1.1460	5.4980	35.1	113.8	
hpdl380g9	Linux	3.10.0-	3493	1.1470	5.7040	33.2	114.5	
hpg9local	Linux	3.10.0-	3493	1.1460	4.6780	31.7	110.5	
hpg9remot	Linux	3.10.0-	3493	1.1450	5.6430	41.0	165.8	
insplocal	Linux	3.10.0-	3602	1.1130	4.5860	18.8	81.3	
inspremot	Linux	3.10.0-	3602	1.1130	3.3400	50.9	195.1	
inspur-nf	Linux	3.10.0-	3602	1.1130	3.3390	18.7	80.6	
inspur-nf	Linux	3.10.0-	3602	1.1120	3.3420	18.7	81.0	
inspur-nf	Linux	3.10.0-	3602	1.1120	3.3410	18.5	81.9	
inspur.NF	Linux	2.6.32-	3012	1.2130	5.2230	19.3	84.5	
inspur.NF	Linux	2.6.32-	3012	1.2130	6.8120	19.3	84.5	
vm.cp	Linux	3.10.0-	3079	1.5880	11.6	65.4	143.6	
vm.cp	Linux	3.10.0-	3079	1.4700	5.7840	30.4	104.4	
vm.inspur	Linux	3.10.0	2885	1.3890	4.3220	22.5	90.9	
vm.inspur	Linux	3.10.0	2885	1.3910	4.2290	21.0	90.8	
vm_dp_vm	Linux	3.10.0-	3279	1.2260	6.4110	30.4	97.8	
vm_dp_vm	Linux	3.10.0-	3279	1.1520	8.8750	29.1	96.8	

For memoey latencies,

- **Inspur NF5270M3 has the best performance on both 2 nodes and local memory access, but has worst performance on remote memory access. (It seems there is the PCB design issue?)**

101

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3 vs. Others

Micro-benchmarking Details

102

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3

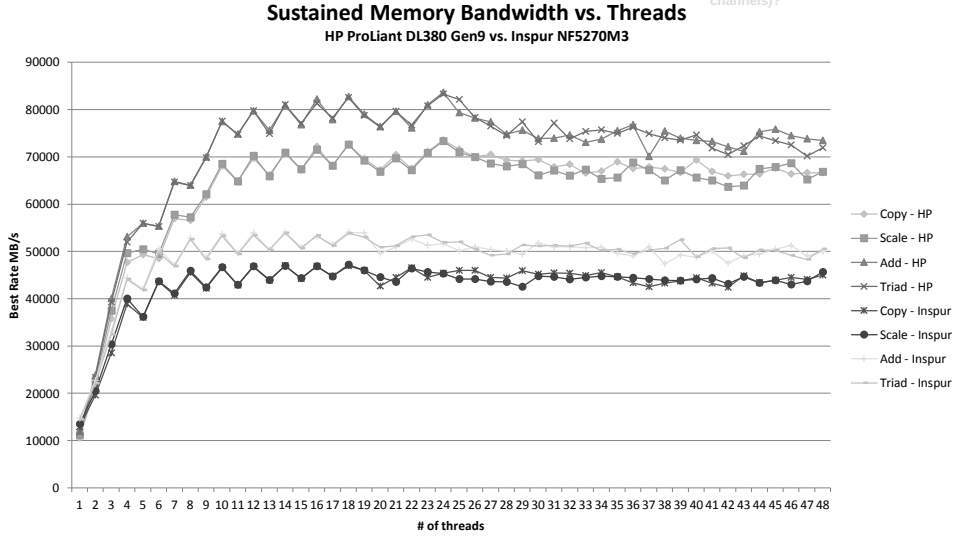
Test Report – STREAM

#define STREAM_TYPE double

Why lines look like this?

HP vs. Inspur on Best Rate. Factors,

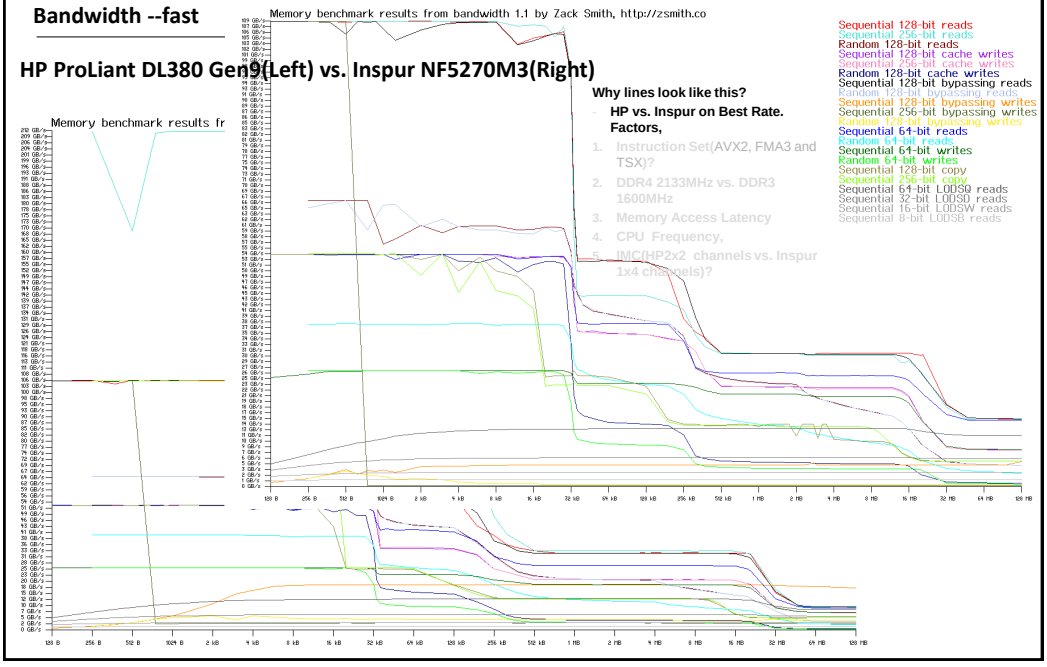
1. DDR4 2133MHz vs. DDR3 1600MHz
2. Memory Access Latency
3. CPU Frequency.
4. IMC(HP2x2 channels vs. Inspur 1x4 channels)?



Test Report – Bandwidth

Bandwidth --fast

HP ProLiant DL380 Gen9(Left) vs. Inspur NF5270M3(Right)



Why lines look like this?

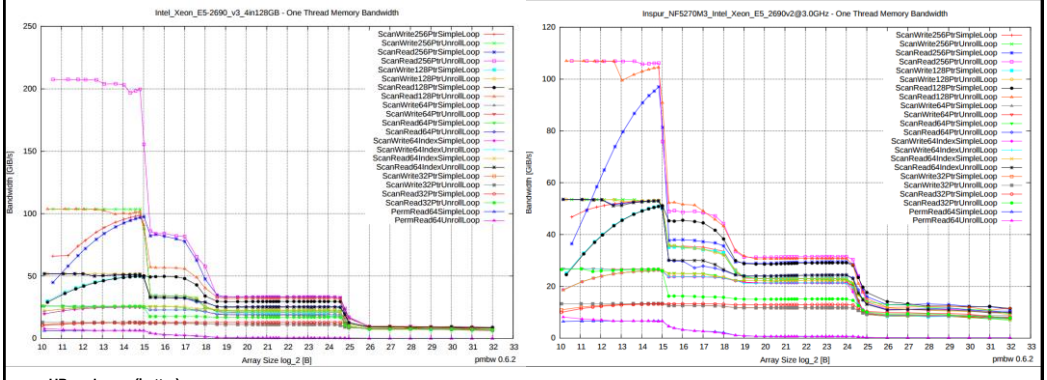
- **HP vs. Inspur on Best Rate. Factors,**
  1. Instruction Set(AVX2, FMA3 and TSX)?
  2. DDR4 2133MHz vs. DDR3 1600MHz
  3. Memory Access Latency
  4. CPU Frequency,
  5. IMC(HP2x2 channels vs. Inspur 1x4 channels)?

Test Report – pmbw

HP vs.Inspur - Single Thread Memory Bandwidth

Why lines look like this?

- **Refer to Bandwidth above**



HP vs. Inspur(better)

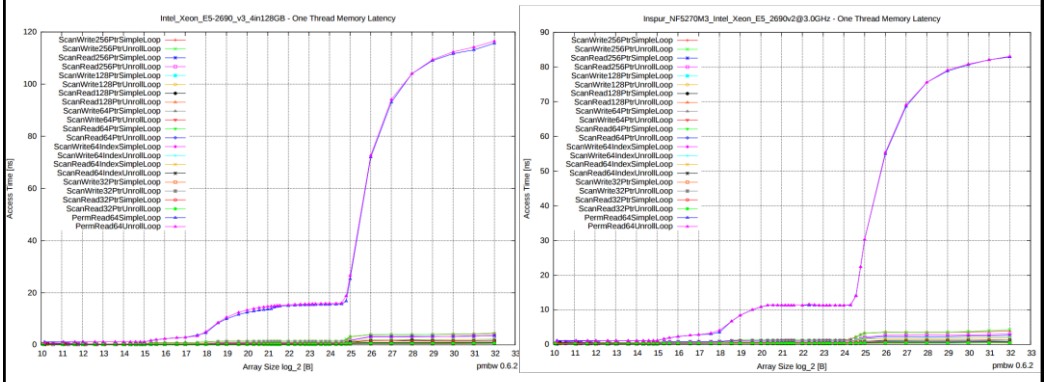
BW of L1 cache: HP

- Instruction Set(AVX2, FMA3 and TSX)?

Test Report – pmbw(cont.)

HP vs.Inspur - Single Thread Memory Bandwidth

Why lines look like this?
- **Refer to Bandwidth above**



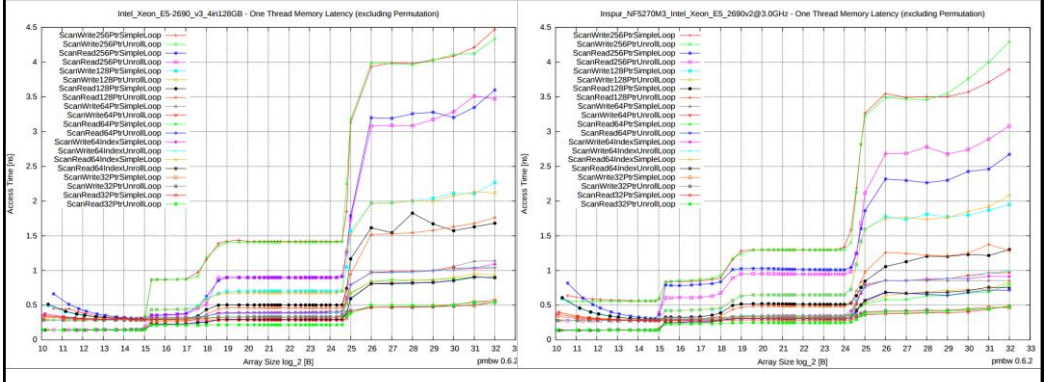
HP vs. Inspur(better)

Access Time: Inspur
- CPU Frequency 3.0GHz
- 1x4 channels Integrated Memory Controller?

Test Report – pmbw(cont.)

HP vs.Inspur - Single Thread Memory Bandwidth

Why lines look like this?
- **Refer to Bandwidth above**



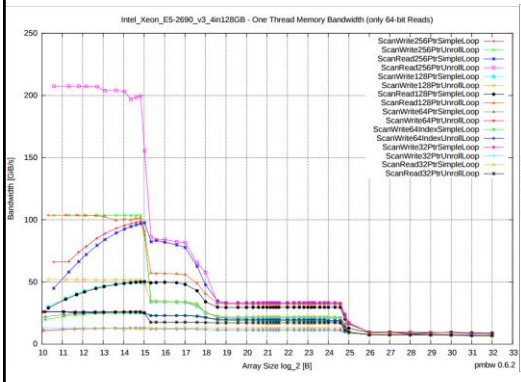
HP vs. Inspur(better)

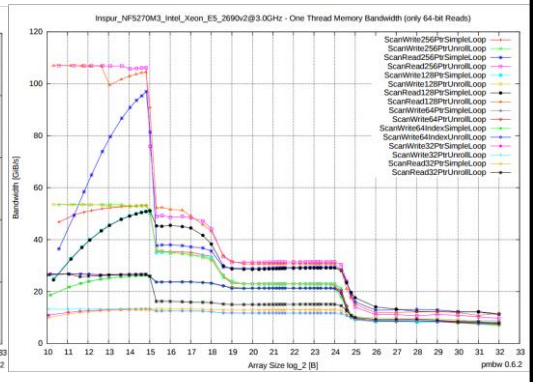
Access Time: Inspur
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

Test Report – pmbw(cont.)

Why lines look like this?
Refer to Bandwidth above

HP vs.Inspur - Single Thread Memory Bandwidth





HP vs. Inspur(better)

BW of L1 cache: HP

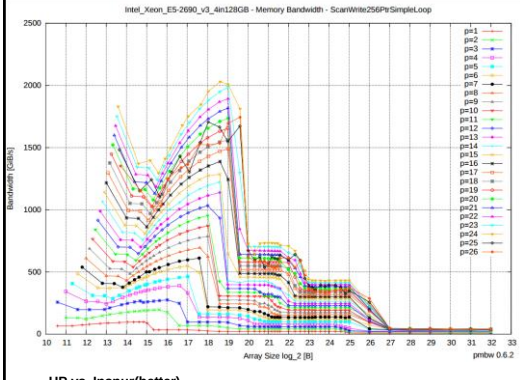
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- channels of IMC?

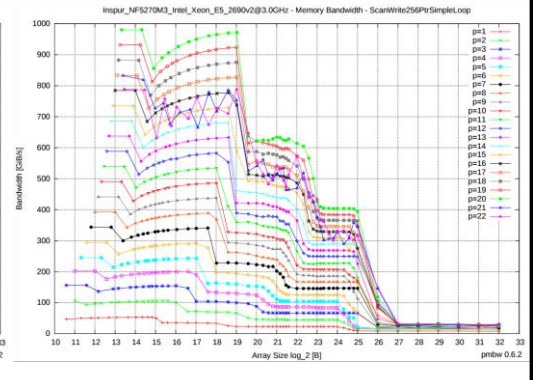
109

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

BW of L1 cache: HP

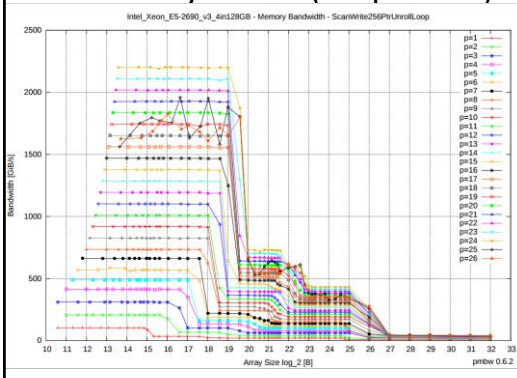
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

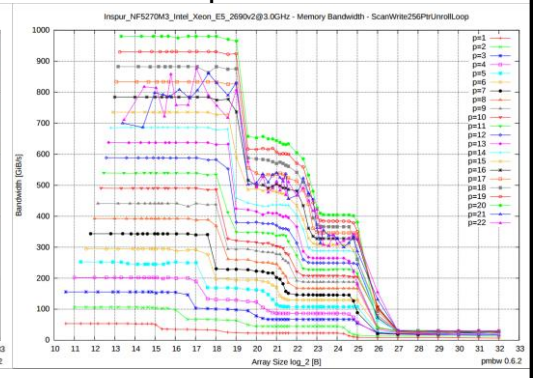
110

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

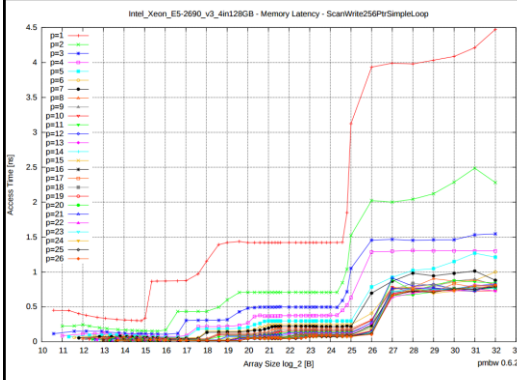
BW of L1 cache: HP

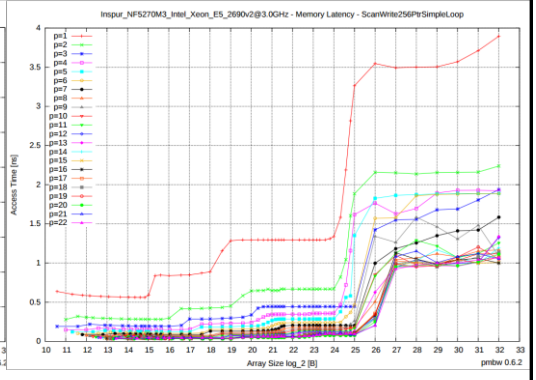
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

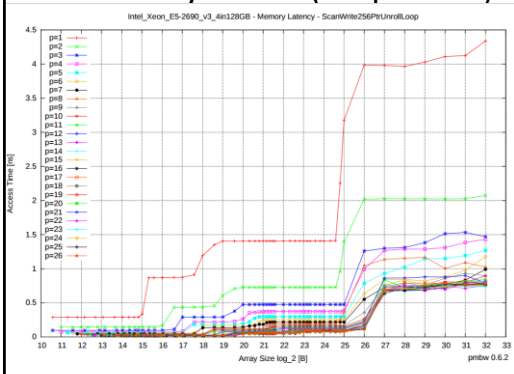
Access Time: Inspur

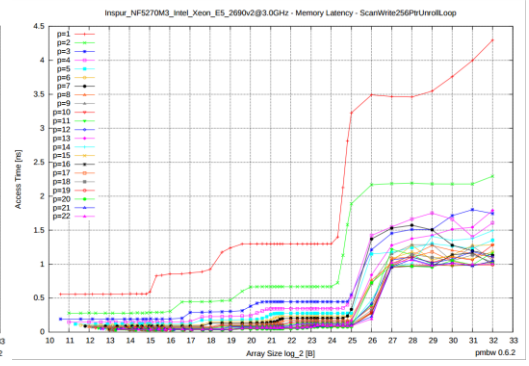
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

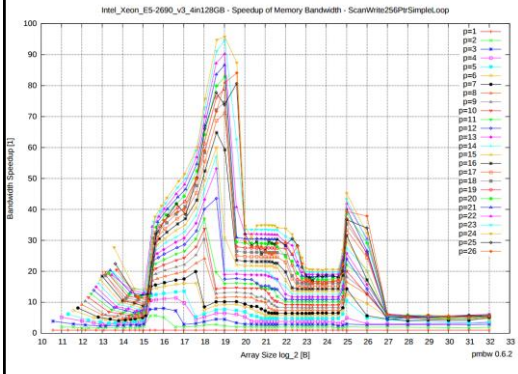
Access Time:

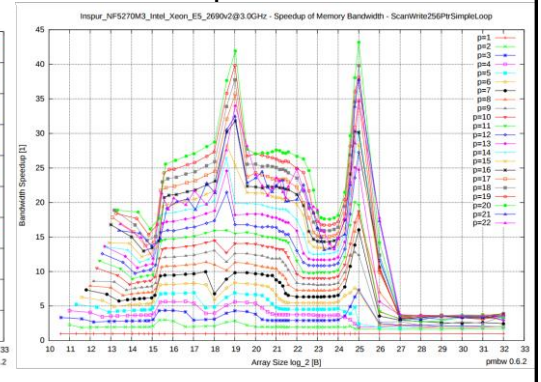
- CPU Frequency 3.0GHz
- 1x4 channels IMC?

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





HP vs. Inspur(better)

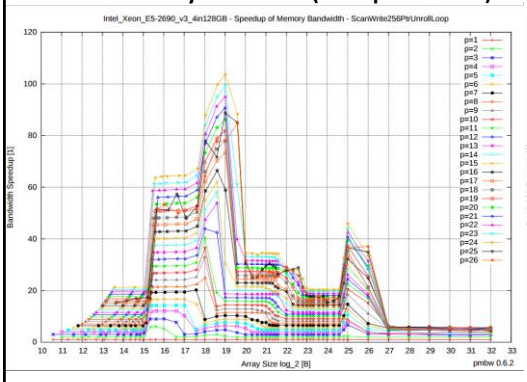
BW of L1 cache: HP

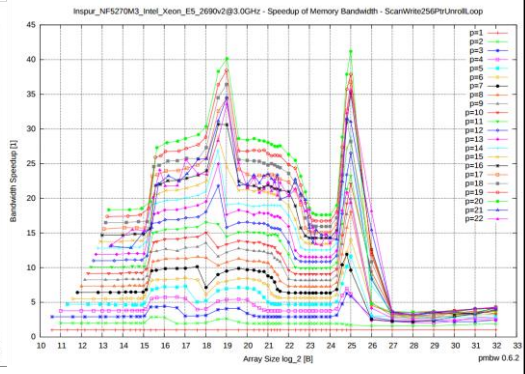
- Instruction Set(AVX2, FMA3 and TSX)?
- CPU Frequency
- # Channels of IMC?
- # cores/Processor

Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll

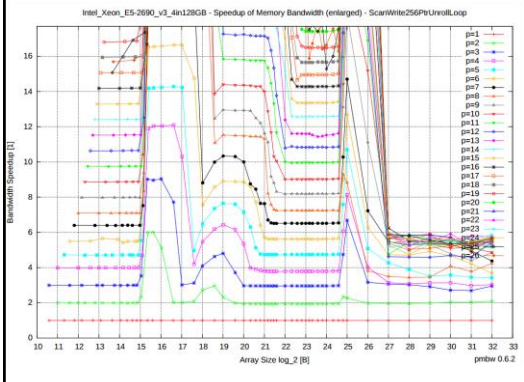


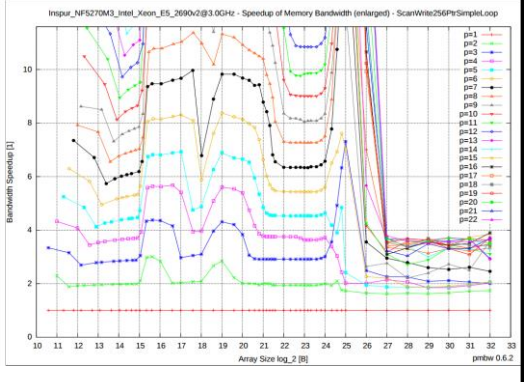


Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll

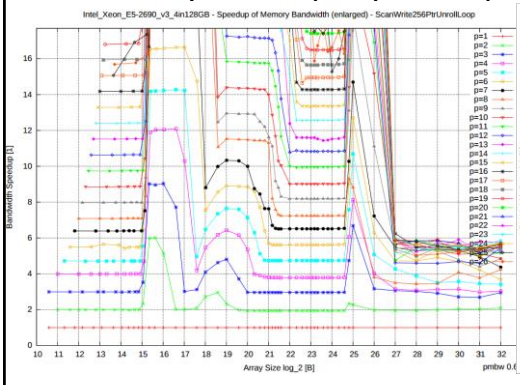


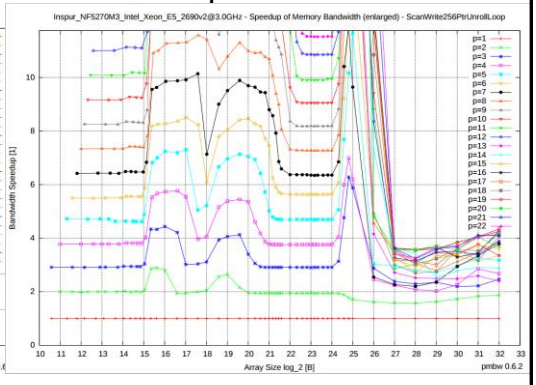


Test Report – pmbw(cont.)

HP vs. Inspur

Parallel Memory Bandwidth(Multiple threads)–ScanWrite256Ptr_Simple vs. Unroll





Test Report – lmbench

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3

```
*Local* Communication bandwidths in MB/s - bigger is better
-----------------------------------------------------------------------------
Host                OS  Pipe AF    TCP  File   Mmap  Bcopy  Bcopy  Mem   Mem
                             UNIX      reread reread (libc) (hand) read write
--------- ------------- ---- ---- ---- ------ ------ ------ ------ ---- -----
hpdl380g9 Linux 3.10.0- 4291 8515 5347 5002.2 5819.7 7793.7 5533.8 8684 6798.
hpdl380g9 Linux 3.10.0- 4336 8638 4894 5997.2 7573.1 7588.6 5617.5 8731 6797.
hpdl380g9 Linux 3.10.0- 4302 8620 5094 6150.6 7861.3 7644.9 5412.7 8716 6810.
hpg9local Linux 3.10.0- 4325 8405 5079 6300.7 8262.5 8057.9 5787.3 8808 7209.
hpg9remot Linux 3.10.0- 4251 7968 5271 5345.4 6539.8 7499.9 4868.2 7191 5521.
insplocal Linux 3.10.0- 4638 9239 5055 8000.1  12.8K 3410.2 6326.7 12.K 8062.
inspremot Linux 3.10.0- 4726 9556 5149 4773.5 5591.4 1452.8 4070.2 6124 5101.
inspur-nf Linux 3.10.0- 4549 9506 5209 5130.0 6336.6 3361.3 6218.8 12.K 8073.
inspur-nf Linux 3.10.0- 4622 9510 5202 7208.6 9540.3 3363.9 6491.6 12.K 8081.
inspur-nf Linux 3.10.0- 4586 9602 5185 5132.5 9506.2 3363.9 6392.5 12.K 8153.

Memory latencies in nanoseconds - smaller is better
    (WARNING - may not be correct, check graphs)
-----------------------------------------------------------------------------
Host                 OS   Mhz   L1 $   L2 $    Main mem    Rand mem    Guesses
--------- ------------- ----   ----   ----    --------    --------    -------
hpdl380g9 Linux 3.10.0-  3493 1.1470 5.7150    35.0        115.1
hpdl380g9 Linux 3.10.0-  3493 1.1460 5.4980    35.1        113.8
hpdl380g9 Linux 3.10.0-  3493 1.1470 5.7040    33.2        114.5
hpg9local Linux 3.10.0-  3493 1.1460 4.6780    31.7        110.5
hpg9remot Linux 3.10.0-  3493 1.1450 5.6430    41.0        165.8
insplocal Linux 3.10.0-  3602 1.1130 4.5860    18.8         81.3
inspremot Linux 3.10.0-  3602 1.1130 3.3400    50.9        195.1
inspur-nf Linux 3.10.0-  3602 1.1130 3.3390    18.7         80.6
inspur-nf Linux 3.10.0-  3602 1.1120 3.3420    18.7         81.0
inspur-nf Linux 3.10.0-  3602 1.1120 3.3410    18.5         81.9
```

results.txt

Communication bandwidth

- **HP is better on Mmap and Bcopy**
- **Inspur is better on Mem read and Mem write**
- **For remote memory access, HP is better.**

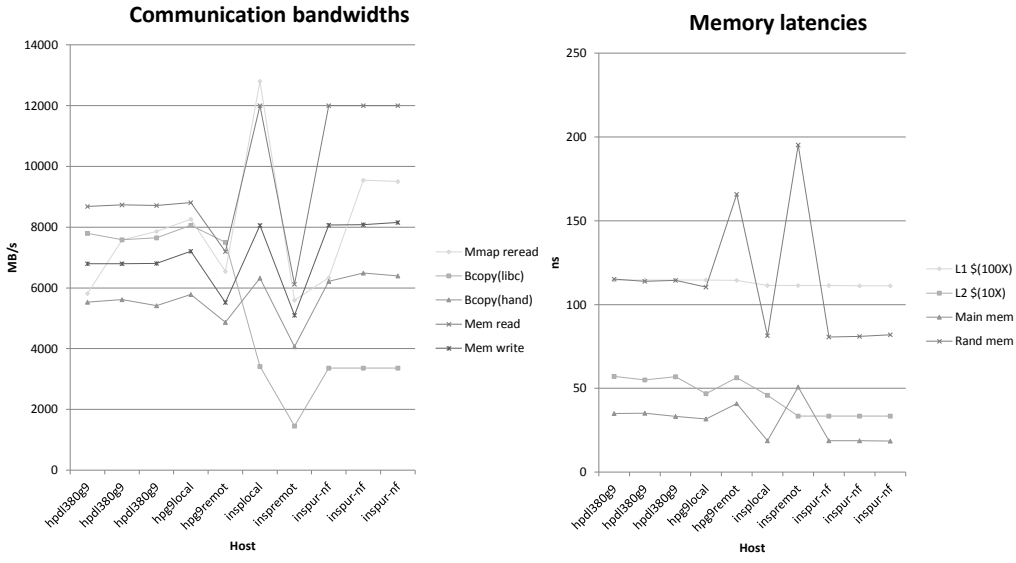
Memoey latencies

- **HP has a bit of latencies more than Inspur on L1 and L2 cache, more latencies on memory subsequential and random memory access**
- **For remote access memory latencies, HP is also better.**

Host		OS	Description	Processor	Memory
hpdl380g9	Linux	3.10.0-			
hpdl380g9	Linux	3.10.0-			
hpdl380g9	Linux	3.10.0-	HP ProLiant	Intel Xeon E5-2690	DDR4 2133MHz
hpdl380g9	Linux	3.10.0-	DL380 Gen9 +	v3 @2.6GHz -	2x4x16GB
hpg9local	Linux	3.10.0-	CentOS7	2x12cores	
hpg9remot	Linux	3.10.0-			
insplocal	Linux	3.10.0-			
inspremot	Linux	3.10.0-	Inspur	Intel Xeon E5-2690	DDR3 1600MHz
inspur-nf	Linux	3.10.0-	NF5270M3 +	v2 @3.0GHz -	2x4x16GB
inspur-nf	Linux	3.10.0-	CentOS7	2x10cores	
inspur-nf	Linux	3.10.0-			

Test Report – lmbench (cont.)

HP ProLiant DL380 Gen9 vs. Inspur NF5270M3



Backup

Why lines look like this?
#

Intel Xeon E5-2690 v2 vs E5-2690 v3

http://www.cpu-world.com/Compare/414/Intel_Xeon_E5-2690_v2_vs_Intel_Xeon_E5-2690_v3.html