Memory Bandwidth and Latency Analysis: AMD Rome, Intel Cascade Lake and Ice Lake Servers

Vamsi Sripathi[*]

September 2020

Contents

List of Figures

[*]Intel Corp. - IAGS/DXPS/DEE/TCE/AWE

List of Tables

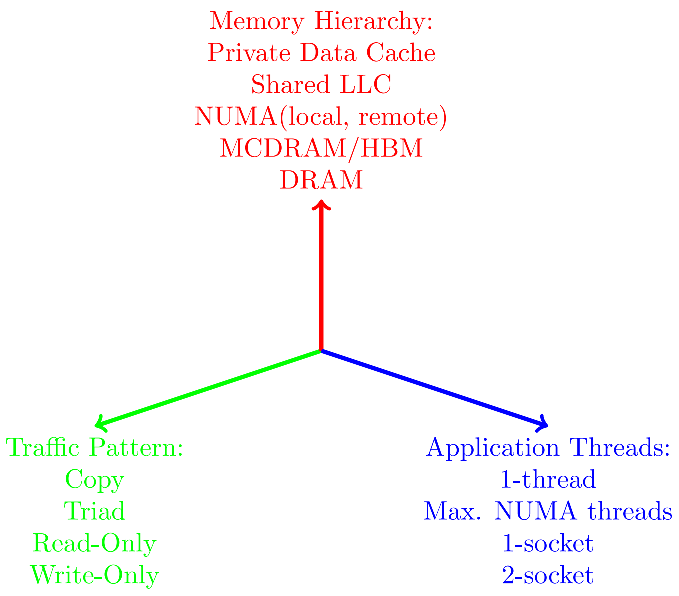


Figure 1: Memory sub-system study

1 Introduction

As Moore's law pushes the compute capabilities, it's imperative to study the bandwidth and latency characteristics of the memory sub-system in order to design a well-balanced CPU architecture. Figure-1 shows one of the approaches to evaluate the memory hierarchy where each of the axis represents a control variable that impacts the observed performance, namely –

1. Memory Hierarchy: Modern CPU's consists of different layers of memory pools. These include multiple levels of on-chip CPU data caches, on-chip multi-channel DRAM (MCDRAM), off-chip DDR. In addition, many CPU's support non-uniform memory access (NUMA) which acts as another memory tier.

2. Application Threads: CPU vendors have taken different architectural routes (e.g., chiplet, large mesh) in scaling the number of cores. As the number of cores in modern CPU's increase, the cost of implementing cache coherency protocols get expensive. On these systems, the number of application threads employed and how these threads are pinned to underlying cpu cores gives an ability to understand the strengths and weaknesses of a CPU architecture.

3. Traffic Pattern: The type of memory traffic generated play a defining role in determining the performance. After all, the memory bandwidth requirements of a dense matrix-matrix multiplication is vastly different from a stencil kernel. In addition, the cache write policies also impact performance depending upon whether a non-temporal store instruction is used in writing back the date to main memory.

1.1 Scope of Study

This report aims to characterize the memory bandwidth and latency for the following 3 CPU architectures:

1. AMD Rome (2nd generation EPYC Processors)
2. Intel Cascade Lake (2nd generation Xeon Scalable Processors)
3. Intel Ice Lake (3rd generation Xeon Scalable Processors)

Since memory bandwidth and latency is a broad topic, we will limit the scope of this study to the following aspects –

1. More emphasis on DDR memory bandwidth followed by cache bandwidth and latency.
2. Synthetic kernels consisting of contiguous access pattern.

In the following sections, we will describe the experiments setup and benchmark methodology. This is followed by providing the DRAM memory bandwidth results for different traffic patterns, thread configurations, type of store instructions and NUMA domain. Next, we will provide the cache bandwidth and latency results.

2 Experiments Setup

2.1 Hardware Platforms

The details of the CPU configuration benchmarked in this study are listed in table-1. It is important to note that the CPU's have different memory channels, memory speed, number of cores etc. However, they are the top end of the SKU's for the corresponding platform and would reflect the best possible performance as observed by end users. In the rest of the document, we will refer to the CPU platforms by their corresponding code-names.

	AMD Epyc 7742	Intel Xeon 8268	Intel ICX XCC
Code name	Rome	Cascade Lake (CLX)	Ice Lake (ICX)
Number of sockets	2	2	2
Number of cores per socket	32	24	32
Total number of cores	64	48	64
LLC per socket	256 MB	35.7 MB	48.3 MB
Total memory	264 GB	197 GB	264 GB
Memory type	DDR4 @ 3200 MT/s	DDR4 @ 2933 MT/s	DDR4 @ 3200 MT/s
Memory channels per socket	8	6	8
Theoretical Peak B/W per socket	204.8 GB/s	140.8 GB/s	204.8 GB/s
Theoretical Peak B/W	409.6 GB/s	281.6 GB/s	409.6 GB/s

Table 1: Hardware Platforms

The theoretical peak memory bandwidth is calculated as:

$$Peak\ B/W\ (GB/s) = x\ GT/s\ \times\ 8\ bytes\_per\_channel\ \times\ n\ channels\_per\_socket \times m\ sockets$$

2.2 STREAM Benchmark

STREAM is a widely used standard benchmark to characterize the memory performance. It consists of the following 4 kernels – Copy, Scale, Add, Triad. The characteristics of these kernels are listed in table-2. As can be observed from the table, Scale and Add process the same number of bytes as Copy and Triad respectively and only differ in the floating-point operations. The differences in floating-point calculations is insignificant as these operations are memory-bound. Hence, we do not show the performance of these two kernels in the remaining part of the report.

To cover additional traffic patterns, we developed two more kernels representing 100% read (Reduce) and 100% write traffic (Fill). The Reduce kernel performs a vector reduction by an *add* opeartor and the Fill kernel writes a constant scalar value to all elements of a buffer. The source code for these kernels can be accessed at the following internal GIT repository https://gitlab.devtools.intel.com/vsripath/stream/

2.2.1 Target ISA

The ISA used determines how many number of bytes are transferred to/from memory from/to CPU registers in a single vector instruction. AVX2 and AVX512F vector instructions operate on 256b and 512b wide CPU registers respectively and hence transfer 32B and 64B per load/store instruction respectively. STREAM benchmark is compiled with Intel C Compiler with the relevant flags to generate the optimal instruction generation for the target processor. The highest ISA supported on AMD Rome and Intel CLX, ICX is AVX2 and AVX512F respectively. Hence, AVX2 and AVX512F instructions are generated in the binary that is executed on AMD Rome and Intel CLX, ICX respectively. Refer to the makefile at the aforementioned GIT source repository for all the Compiler flags used in building the binaries used in this study.

2.2.2 Cache Write Policy: vmovpd vs vmovntpd

A CPU caches' write policy determines how and when the data in cache is written back to main memory. Usually, there are two write policies – *write-through and write-back*. Both AMD Rome and Intel CLX, ICX use *write-back (WB)* cache policy. In WB policy, when the CPU modifies data, it's only updated in the cache and the write to the main memory is postponed until the cache line containing the updated data is evicted. Hence, a read-miss on a WB cache may require two memory accesses – if modified, write the contents of the to-be evicted cache line to main memory and then fetch the requested data from main memory. For write-misses, they can be two approaches – *write-allocate, no-write allocate*. In *write-allocate* policy, a store that misses the cache first fetches the cache line containing the data from main memory. And then when combined with the *write-back* policy, results in an additional memory access to write the updated data back to main memory.

If we look at the STREAM kernels, we observe the following two characteristics –

- Full Cache Line update: There are no partial cache line modifications done by the kernels.
- Non-Temporal Locality: The data that is written is never accessed/read again. Hence, there is no need to keep the data in cache.

Considering these aspects of STREAM kernels, we can observe that the *write-allocate* policy introduces an unnecessary overhead of fetching the data to cache. This introduces additional constrains on memory bandwidth when all the cores are actively writing data to main memory. To alleviate this, we can use an alternative store instruction called *non-temporal* store (vmovntpd) which by-passes the cache and directly writes to main memory.

In order to understand the impact of type of store instruction on memory bandwidth performance, STREAM benchmark is compiled in the following two configurations –

- Regular Stores: This configuration uses regular stores (vmovpd) that generates two memory accesses on a write-miss. We will refer to this configuration as Read-For-Ownership (RFO) in the rest of the report. RFO is a cache coherency operation specifying a read operation with intent to write to that memory address.
- Non-Temporal (NT) Stores: This configuration uses the aforementioned vmovntpd instruction to write back directly to main memory without bringing the data into cache. We will refer to this configuration as NT in the rest of the report.

Kernel	Op	Bytes Read (RFO)	Bytes Read (NT)	Bytes Written	FLOPs
COPY	$a[i] = b[i]$	16	8	8	0
SCALE	$a[i] = scalar \times b[i]$	16	8	8	1
ADD	$c[i] = a[i] + b[i]$	24	16	8	1
TRIAD	$c[i] = a[i] + scalar \times b[i]$	24	16	8	2
REDUCE	$sum+ = a[i]$	8	8	0	1
FILL	$a[i] = constant$	8	0	8	0

Table 2: STREAM Kernels

2.3 Intel Memory Latency Checker

For cache bandwidth and latency analysis, we rely on the more versatile Intel Memory Latency Checker.

2.4 Runtime Settings

Table-3 shows the run-time settings used for most of the results presented in this study. Where we depart from these settings, we will explicitly call out the changes in the corresponding section.

	AMD Rome 7742	Intel Xeon 8268	Intel ICX XCC
NUMA per socket	4	1	1
Operating System	CentOS Linux 7 (Core)		
Kernel	3.10.0-1127.18.2.el7.crt1.x86_64		
CPU Turbo Boost	Enabled		
CPU Scaling Governor	userspace		
CPU Scaling Driver	acpi-cpufreq		
Transparent Huge Pages	Enabled		
Intel C Compiler	ICC 19.1.2.254 20200623		
Hyper-Threading	Enabled in HW, but logical threads are not used in benchmarking		
OpenMP Thread Affinity	KMP_AFFINITY = "granularity=fine,compact,1,0"		

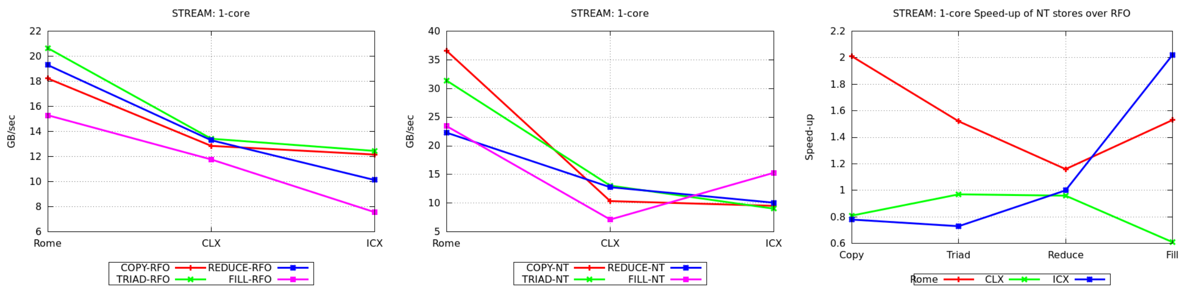
Table 3: Runtime Settings

3 DRAM Bandwidth

In the following sub-sections, we will look at the peak achieved memory bandwidth by different CPU's in 1-core, 1-socket and 2-socket configuration. We will also evaluate the scaling performance within a socket and the impact of NUMA domains and OpenMP thread to CPU core affinity as well. Each sub-section contains performance data to primarily answer the following two questions –

1. How does the different CPU architectures compare against each other?
2. What is the impact of using RFO and NT stores on a given architecture?

3.1 1-Core



(a) RFO (b) NT (c) Speed-up of NT over RFO

Figure 2: STREAM 1-core DRAM bandwidth

3.1.1 Peak Bandwidth with RFO

Figure-2a RFO observations:

1. Rome vs ICX: Even though these two processors have identical DRAM configuration, Rome delivers higher bandwidth across the kernels.
2. ICX vs CLX: Comparing gen-to-gen Xeon performance, there is a slow-down of 37% and 52% for Reduce and Fill kernels respectively on ICX. This is despite the fact that ICX has higher DDR4 memory speed and channels.
3. Table-4 summarizes the speed-up of all kernels.

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	18.22	12.85	12.16	0.67	0.71	0.95
Triad	20.64	13.42	12.45	0.60	0.65	0.93
Reduce	19.29	13.30	10.13	0.53	0.69	0.76
Fill	15.29	11.77	7.57	0.50	0.77	0.64

Table 4: 1-core peak bandwdith: RFO

3.1.2 Peak Bandwidth with NT

Figure-2b NT observations:

1. Rome vs ICX: Similar to the trend observed with RFO stores, Rome delivers higher bandwidth across the kernels with the speed-up over ICX even greater.
2. ICX vs CLX: On ICX, the Fill kernel using NT stores shows marked improved over CLX. And the rest of the kernels exhibit slow-down.
3. Table-5 summarizes the speed-up of all kernels.

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	36.57	10.36	9.53	0.26	0.28	0.92
Triad	31.37	13.06	9.04	0.29	0.42	0.69
Reduce	22.29	12.78	10.08	0.45	0.57	0.79
Fill	23.43	7.15	15.30	0.65	0.31	2.14

Table 5: 1-core peak bandwdith: NT

3.1.3 RFO vs. NT Stores

For the different STREAM kernels, we can come up with the following expected speed-up of NT over RFO –

- Copy: The Copy kernel has a total of 2 (1 read, 1 write) and 3 (2-reads, 1 write) memory-ops when using NT and regular stores respectively. Hence, the ideal speed-up of NT over RFO should be 1.5x.
- Triad: The Triad kernel has a total of 3 (2 reads, 1 write) and 4 (3 reads, 1 write) memory-ops when using NT and regular stores. Hence, the ideal speed-up of NT over RFO should be 1.33x.
- Reduce: The Reduce kernel has 0 write ops and so the performance should remain the same with both NT and regular stores.
- Fill: The Fill kernel has only memory write ops and so the ideal speed-up of NT over RFO should be 2x.

Figure-2c observations:

1. Across the CPU's, the Reduce kernel performance remains identical between NT and RFO configurations since it has no writes to main memory.
2. Rome: As can be expected, NT stores show higher performance than using regular stores which generate RFO requests. Interestingly, the speed-up of Copy kernel with NT stores exceeds the expected speed-up.
3. CLX: None of the kernels shows gains with NT. This indicates that memory bandwidth is not the constraint when using 1 core because RFO generates more memory traffic than NT. This leads to the observation that when the system is not memory bandwidth constrained (as is the case here with 1-thread), the latency of NT stores is greater than regular stores. This could be due to the fact that the hardware prefetchers do a good job of fetching the write-miss requests with regular stores. In contrast, with NT stores the hardware prefetchers have no active role since the data is never to be brought into cache.
4. ICX: Only the Fill kernel using NT stores shows higher performance than regular stores. Find out why?

3.2 1-Socket

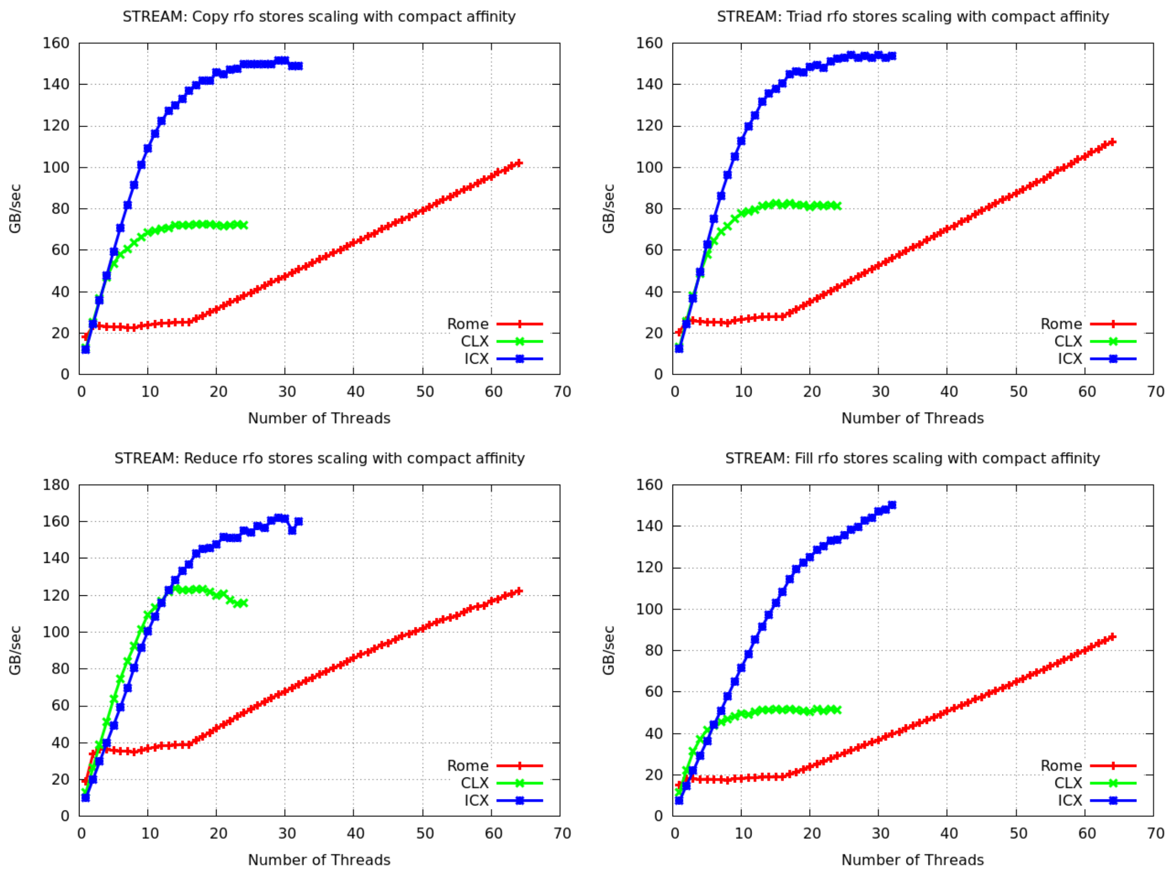


Figure 3: STREAM 1-socket DRAM bandwidth scaling with RFO and Compact pinning

For these experiments, Rome is configured to be in 4 NUMA domains Per Socket (NPS) mode whereas ICX and CLX are configured to have 1 NPS. Also, OpenMP threads are pinned to cores that are closer first i.e., compact mode. We will explore the performance impact of distributed pinning in the next section.

3.2.1 Peak Bandwidth and Scaling with RFO

Figure-3 observations:

- Peak Bandwidth: Using all the cores in the socket
  - Rome vs ICX: Contrary to 1-core performance, ICX beats Rome for all kernels using RFO stores.
  - Rome vs CLX: Rome delivers superior performance for all kernels.
  - ICX vs CLX: Comparing gen-to-gen Xeon performance, ICX delivers significant performance improvements over CLX for all kernels.
  - Table-6 shows the speed-up of all kernels.
- Scaling Performance:
  - Rome: Across the 4 kernels, Rome exhibits similar scaling performance pattern. Since Rome contains 64 cores per socket partitioned into 4 NUMA domains, each NUMA domain contains 16 cores. The data in the figure shows that the performance remains similar from 1 to 16 threads i.e., when threads are populated in NUMA domain-0. After that, the performance increases linearly with thread count and the gains from 16 to 64 threads show ideal scaling behavior.
  - CLX, ICX: In contrast to Rome, both CLX and ICX reach close to peak bandwidth using much less number of threads. In addition, CLX shows higher performance at some thread counts for Reduce and Fill kernels.

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	102.08	71.92	148.85	1.46	0.70	2.07
Triad	112.16	81.35	153.71	1.37	0.73	1.89
Reduce	122.46	115.73	159.95	1.31	0.95	1.38
Fill	86.61	51.38	150.26	1.73	0.59	2.92

Table 6: 1-socket peak bandwdith: RFO

3.2.2 Peak Bandwidth and Scaling with NT

Figure-4 observations:

- Peak Bandwidth: Using all the cores in the socket
  - Rome vs ICX: Contrary to RFO, ICX delivers similar performance as Rome, with Triad showing 27% gains and Fill showing slow-down.
  - Rome vs CLX: Rome delivers superior performance for all kernels.
  - ICX vs CLX: Comparing gen-to-gen Xeon performance, ICX delivers 40-70% performance improvements over CLX for all kernels.
  - Table-7 shows the speed-up of all kernels.
- Scaling Performance: The scaling profile looks similar to RFO except for the following exceptions –
  - Rome: The Fill kernel shows gains with thread count starting at 8 instead of 16 as observed with RFO stores.
  - ICX: The Fill kernel loses performance when using all the threads in the socket. This results in reduction of about 15% peak bandwidth for this kernel.

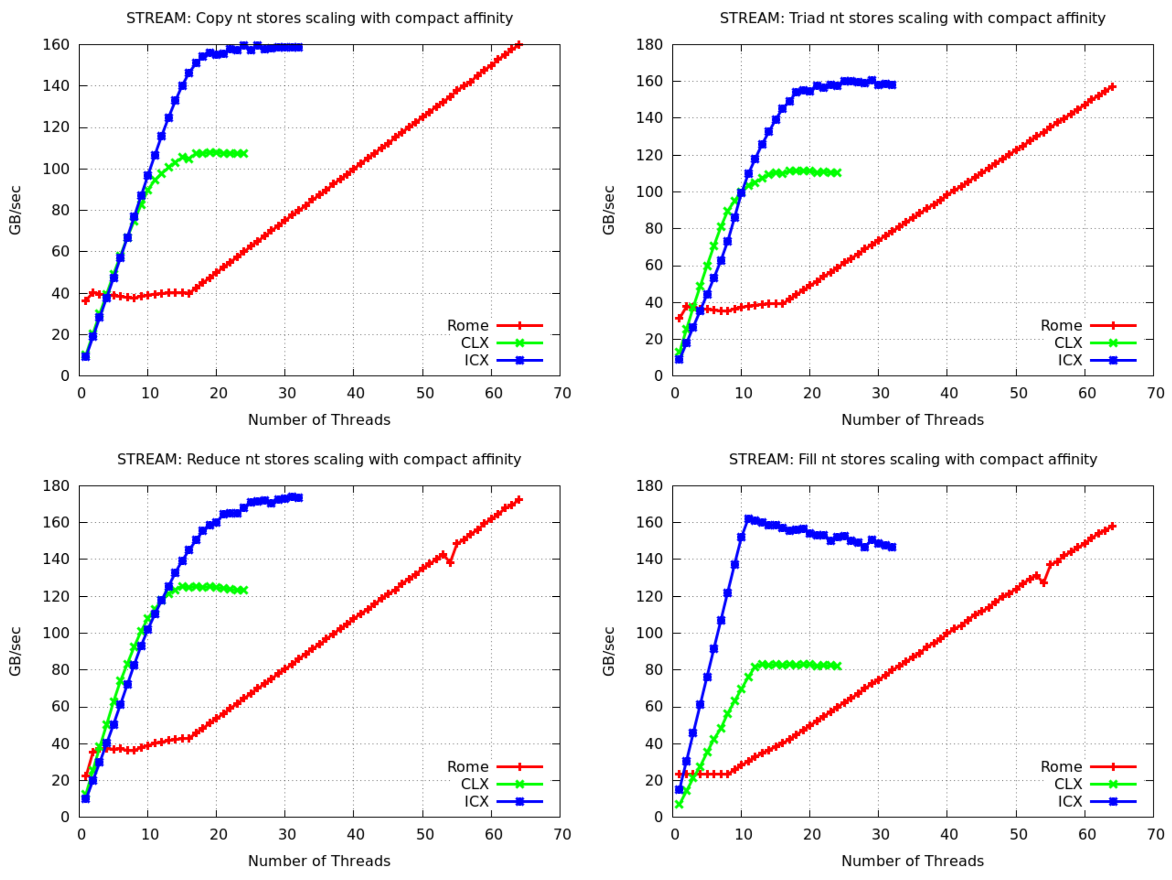


Figure 4: STREAM 1-socket DRAM bandwidth scaling with NT Stores and Compact pinning

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	159.77	107.55	158.43	0.99	0.67	1.47
Triad	157.31	110.45	158.21	1.01	0.70	1.43
Reduce	172.57	123.45	173.41	1.00	0.72	1.40
Fill	158.14	81.99	146.69	0.93	0.52	1.79

Table 7: 1-socket peak bandwdith: NT

3.2.3 RFO vs. NT Stores

1. Copy: As mentioned in the previous section, the ideal speed-up of NT over RFO for Copy kernel should be 1.5x. Rome matches with the expected ideal speed-up. CLX reaches the 1.5x speed-up at higher thread count whereas ICX shows far lesser gains with NT stores. Also, to be noted is that both CLX and ICX shows performance degradation with NT stores with lower thread counts which isn't the case with Rome.

2. Triad: The ideal speed-up of NT over RFO for Triad should be 1.33x. Rome reaches the ideal speed-up at some thread counts and then appears to exhibit lower gains. CLX shows expected gains at higher thread counts, whereas ICX shows maximum gains of about 10% with NT stores. Similar to the Copy kernel, both CLX and ICX exhibit performance degradation with NT stores at lower thread counts.

3. Reduce: The performance of Reduce kernel should remain identical with both NT and regular stores. However, the Reduce kernel on Rome shows upto 40% performance gains with NT stores. Deeper investigation of this kernel code generation revealed that the Intel compiler does aggressive unrolling when using regular stores. This results in more in-flight loads and appears to be the root-cause for the slowdown with regular stores. Interestingly, even though the Intel Compiler does the same aggressive unrolling with regular stores on CLX and ICX, these two architectures seems to be more resilient to additional in-flight loads and do not result in lower performance.

4. Fill: The ideal speed-up of NT over RFO should be 2x. Rome reaches the expected gains once the thread count is greater than the number of cores of NUMA domain. CLX and ICX exhibit different behavior for this kernel with ICX showing significant performance loss as thread count increases.

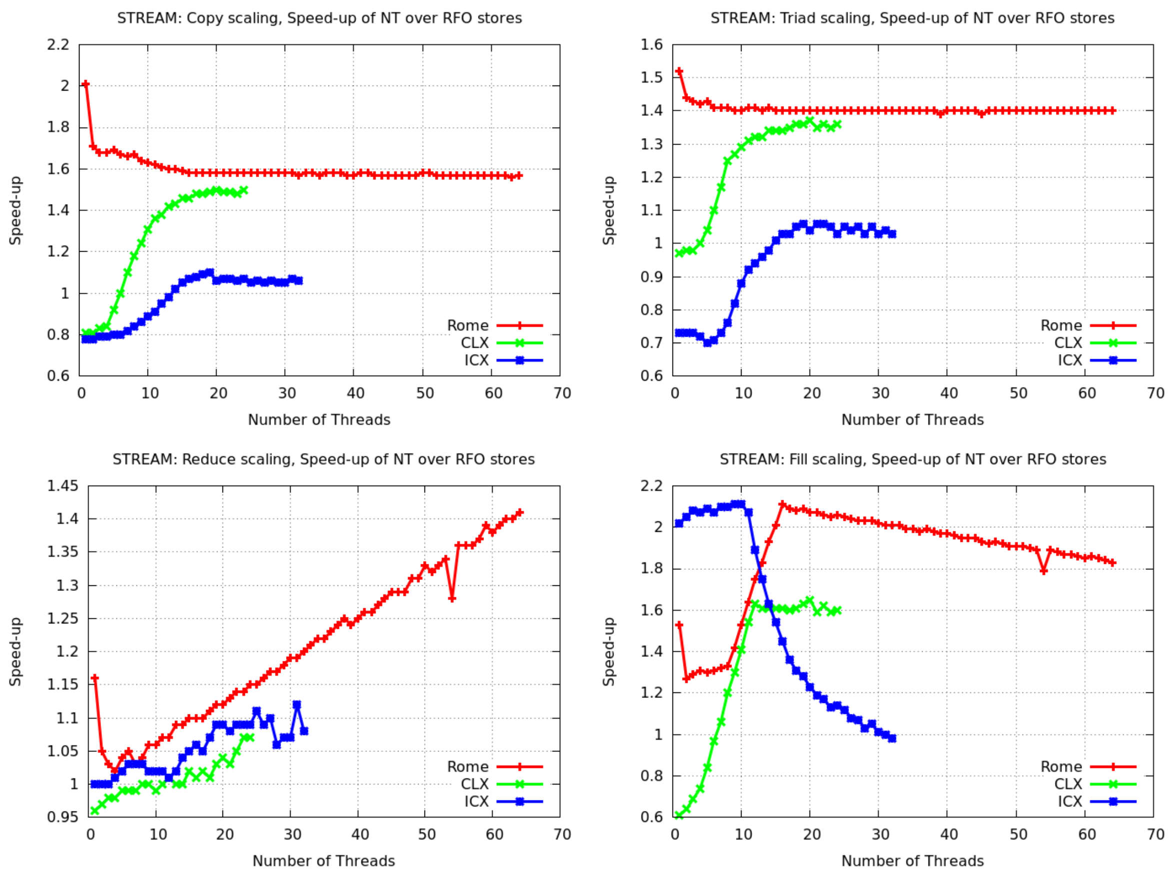


Figure 5: STREAM 1-socket DRAM bandwidth scaling, speed-up of NT over RFO Stores

3.2.4 Compact vs Distributed Pinning

The experiments with thread affinity settings make for an interesting performance study only when there are multiple NUMA domains in a socket. Since only Rome is configured to have 4 NUMA domains per socket in our setup, we will limit these experiments to Rome architecture. As mentioned earlier, Compact pinning refers to thread affinity scheme where OpenMP threads are pinned to cores that are closer to each other. In distributed pinning scheme, we affinitize the OpenMP threads to spread across the NUMA domains first. For e.g, with 4 threads this translates to using a single core on each of the 4 NUMA domains in a socket.

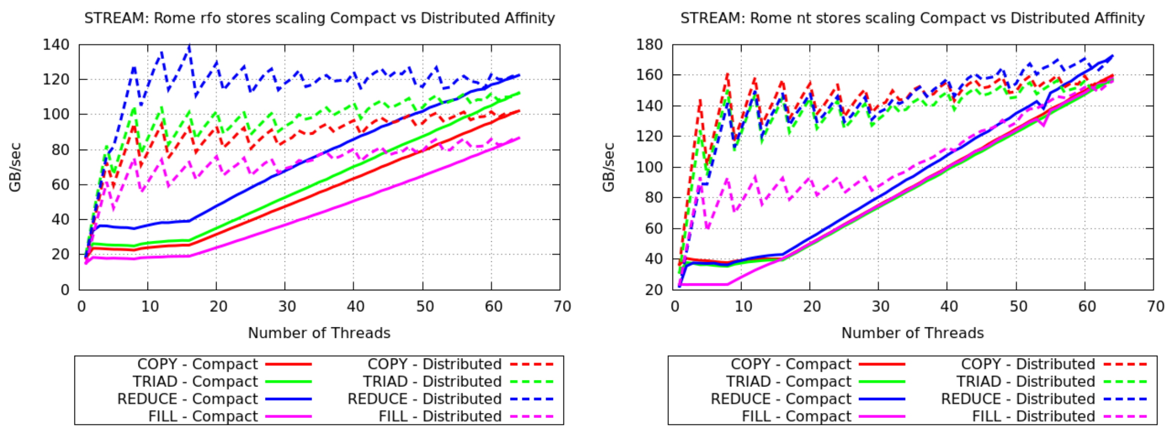


Figure 6: STREAM 1-socket DRAM bandwidth: Compact vs Distributed Pinning

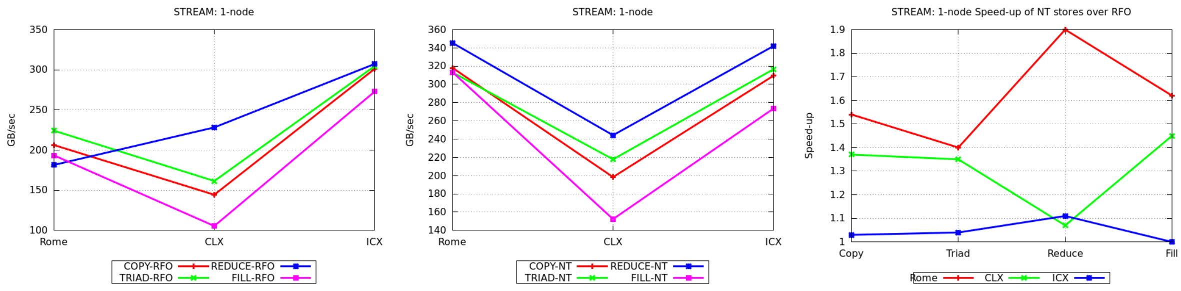
Figure-6 observations:

- For both RFO and NT, close to peak bandwidth is achieved with fewer threads in distributed pinning when compared to compact mode.
- RFO: The Reduce kernel with distributed affinity reaches peak bandwidth at fewer threads and then loses

performance when using all the cores in the socket.

- NT: When compared to the rest of the kernels, the Fill kernel in distributed pinning shows much less achieved bandwidth. Once the number of threads reaches more than 32, the performance is only marginally better than compact pinning.

3.3 2-Socket



(a) RFO (b) NT (c) Speed-up of NT over RFO

Figure 7: STREAM 2-socket DRAM bandwidth

Since each socket has it's own set of memory controllers, DRAM bandwidth for 2-socket scales by a factor of 2 compared to 1-socket performance. There are no unique observations when it comes to 2-socket dataand all the points made in the 1-socket section are valid here as well.

3.3.1 Peak Bandwidth with RFO

Figure-7 shows the peak achieved bandwidth when all cores in a 2 socket system are accessing data to/from DRAM. Table-8, 9 show the corresponding speed-up for each of the architecture with RFO and NT stores respectively..

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	206.17	144.30	301.02	1.46	0.70	2.09
Triad	224.09	161.42	304.43	1.36	0.72	1.89
Reduce	181.47	228.17	307.14	1.69	1.26	1.35
Fill	193.33	105.40	272.65	1.41	0.55	2.59

Table 8: 2-socket peak bandwdith: RFO

3.3.2 Peak Bandwidth with NT

Figure-7 shows the peak achieved bandwidth using NT stores.

Kernel	Rome	CLX	ICX	ICX/Rome	CLX/Rome	ICX/CLX
Copy	318.01	198.35	309.48	0.97	0.62	1.56
Triad	312.99	217.81	316.58	1.01	0.70	1.45
Reduce	345.37	244.26	341.96	0.99	0.71	1.40
Fill	313.32	152.32	273.49	0.87	0.49	1.80

Table 9: 2-socket peak bandwdith: NT

3.4 NUMA

3.4.1 1-Core, 1-Socket: RFO

3.4.2 1-Core, 1-Socket: NT

3.5 Sub-NUMA Clustering

CLX supports SNC-2, whereas ICX supports SNC-2, 4. To be filled when performance data with SNC-2, SNC-4.

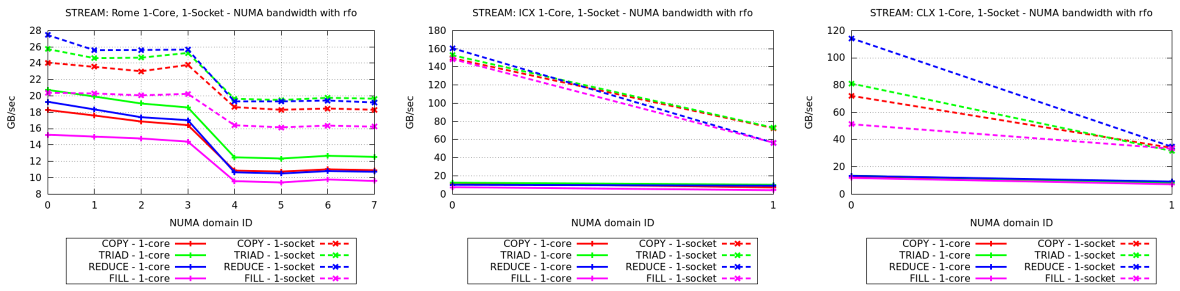


Figure 8: STREAM 1-Core, 1-socket NUMA bandwidth with RFO

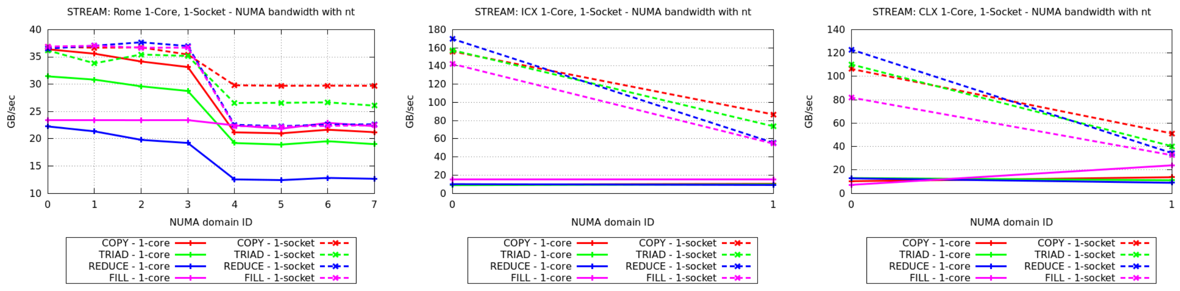


Figure 9: STREAM 1-Core, 1-socket NUMA bandwidth with NT