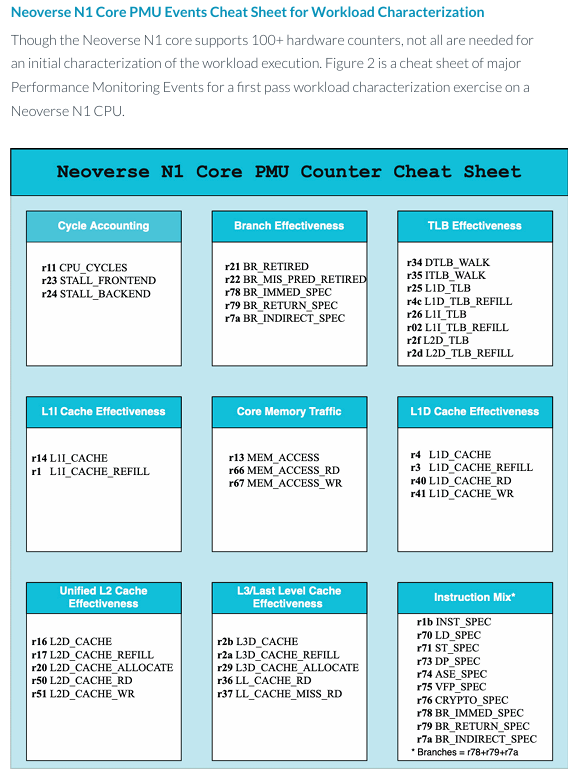
ARM PMU Whitepaper

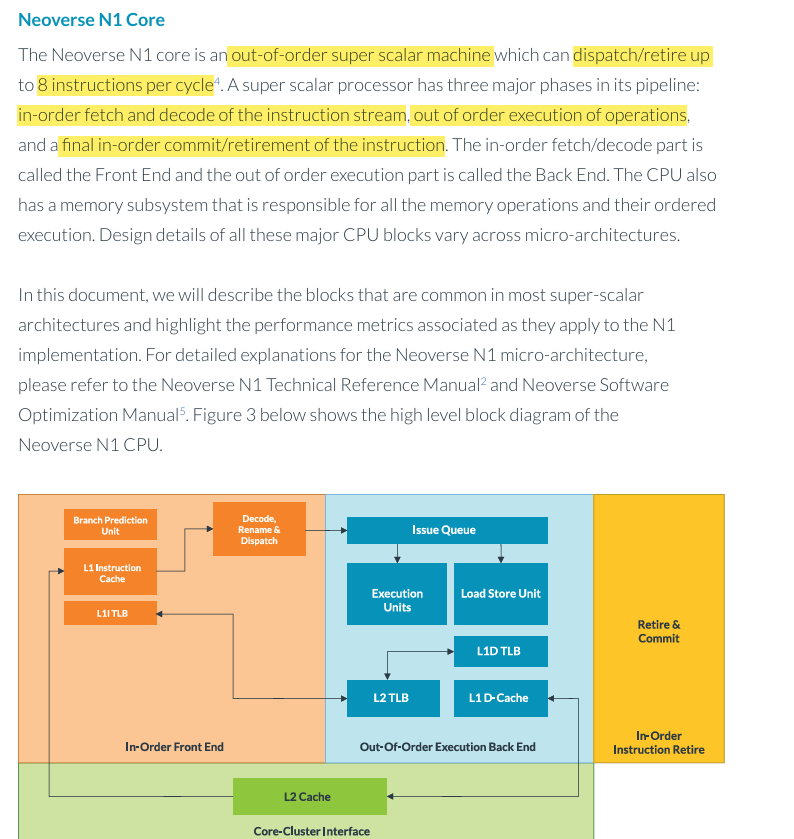
As part of Learning when reading the ARM Whitepaper on “Arm Neoverse N1 Core: Performance Analysis Methodology”. I made few points as part of understanding listing them down.

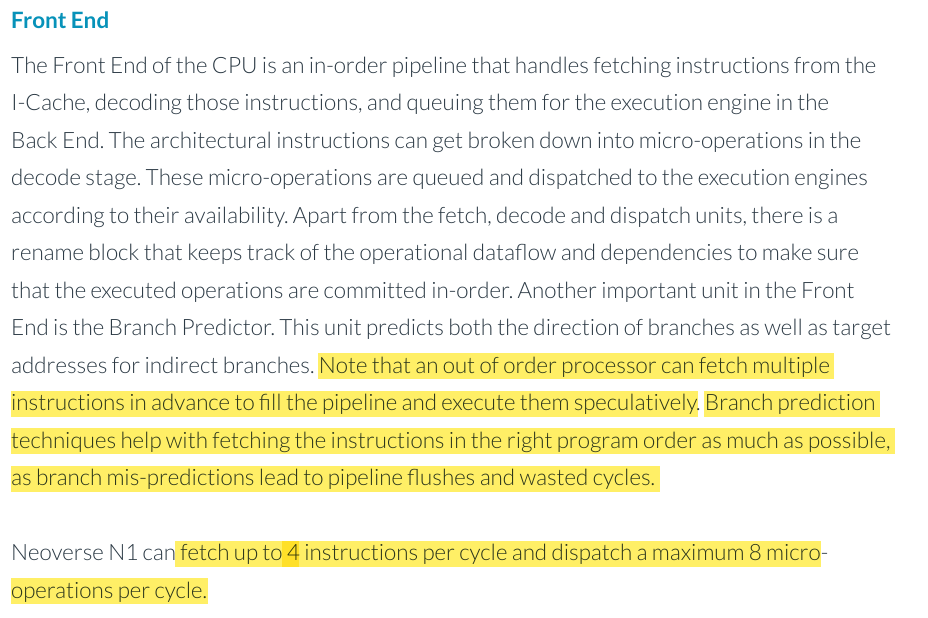
Links :

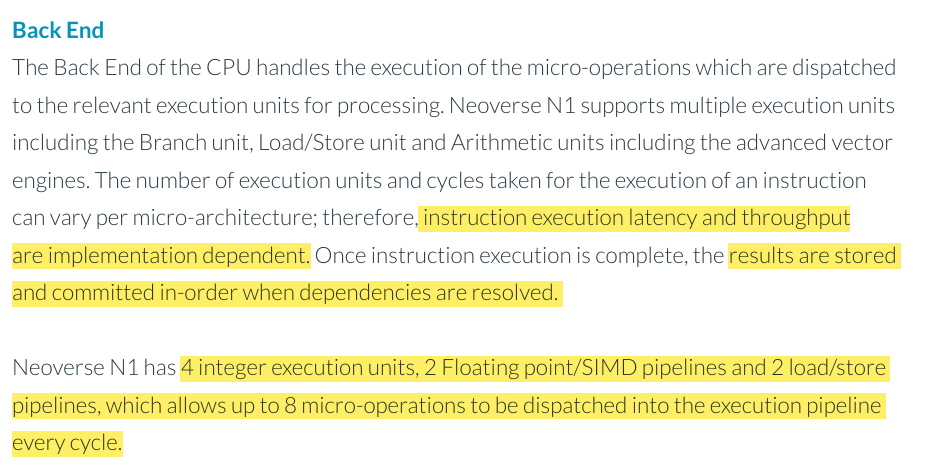
1. <https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/neoverse-n1-core-performance-v2.pdf>
2. <https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/neoverse-v1-core-performance-analysis.pdf>

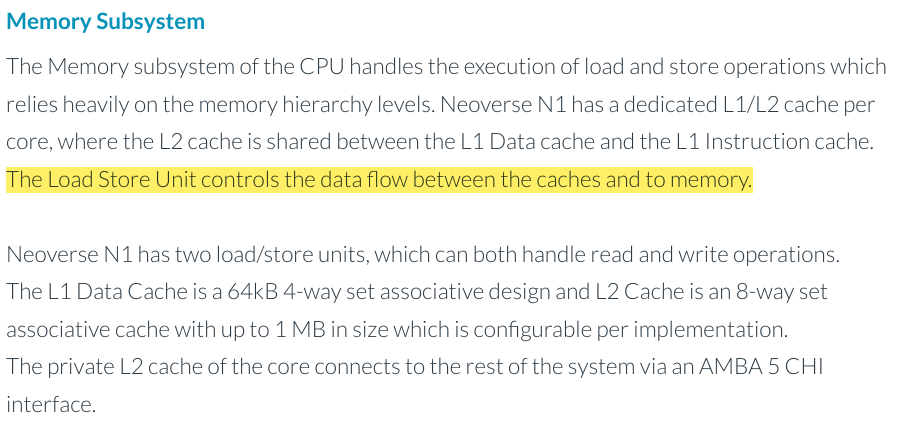
The PMU Metrics for CPU :

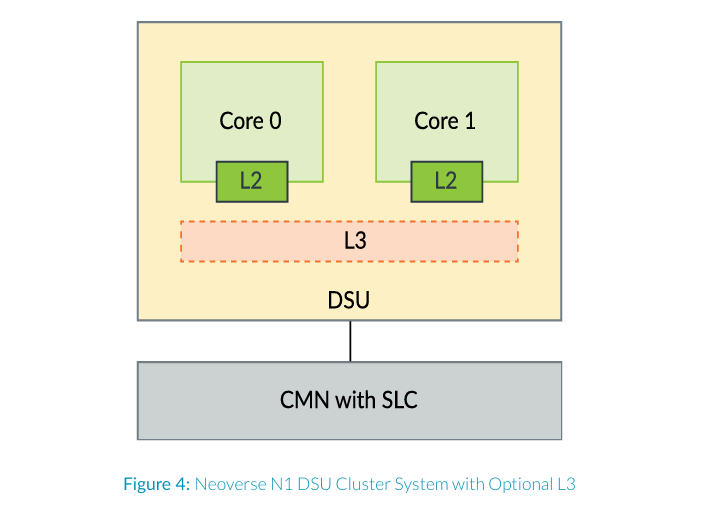
The Neoverse CPU Architecture :

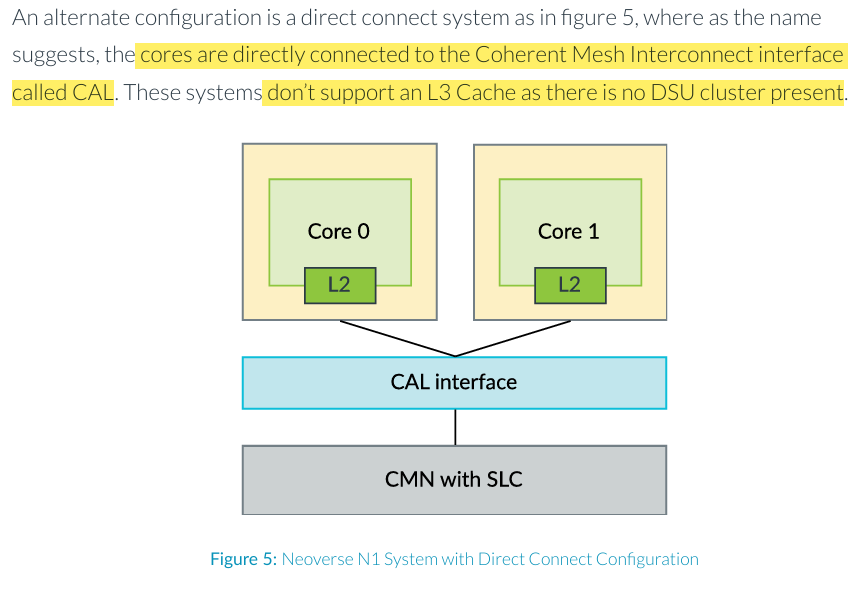


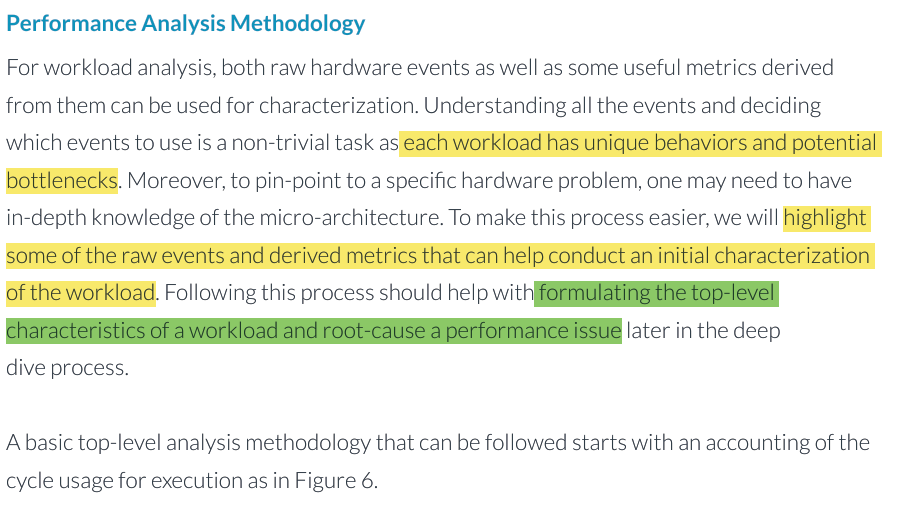


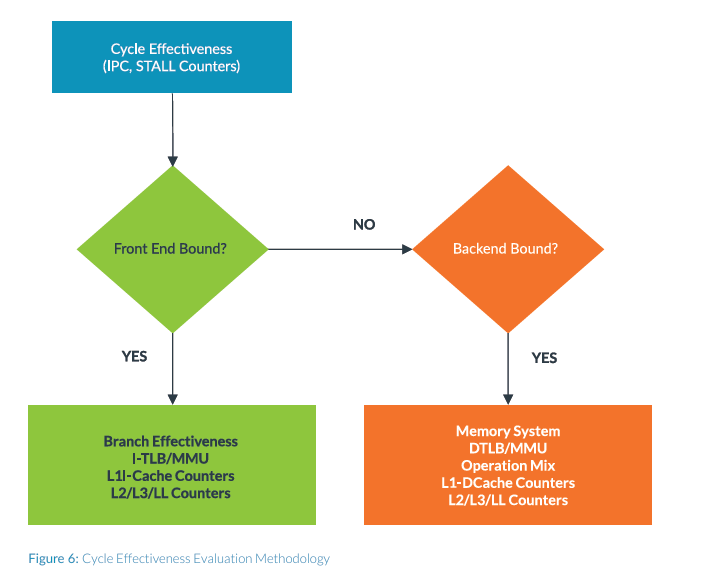


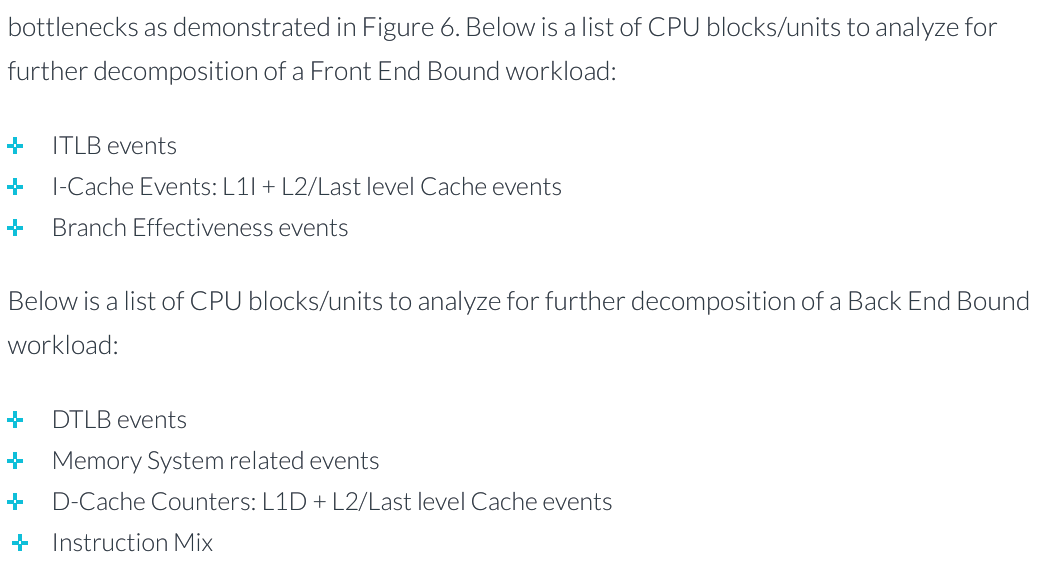


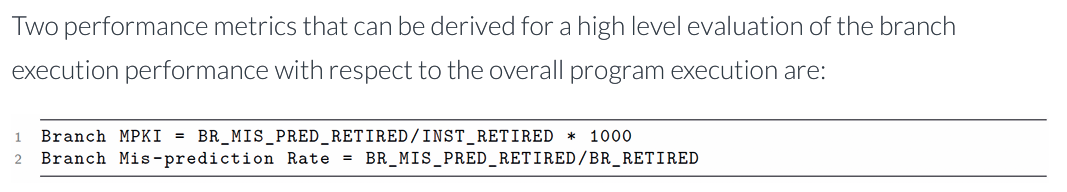


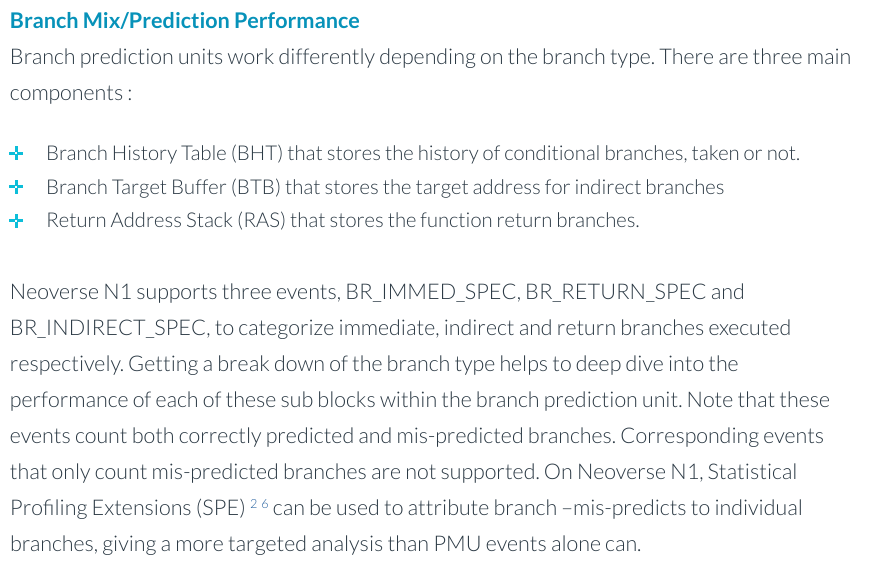


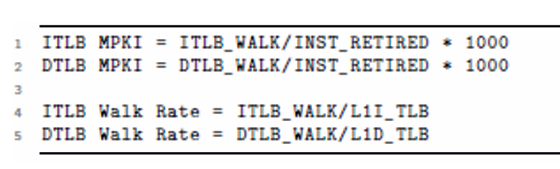


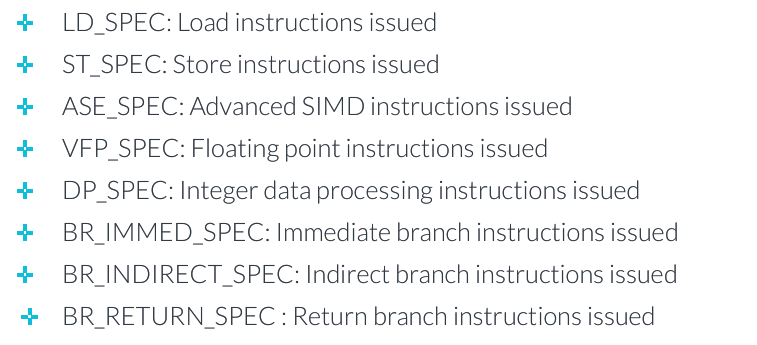


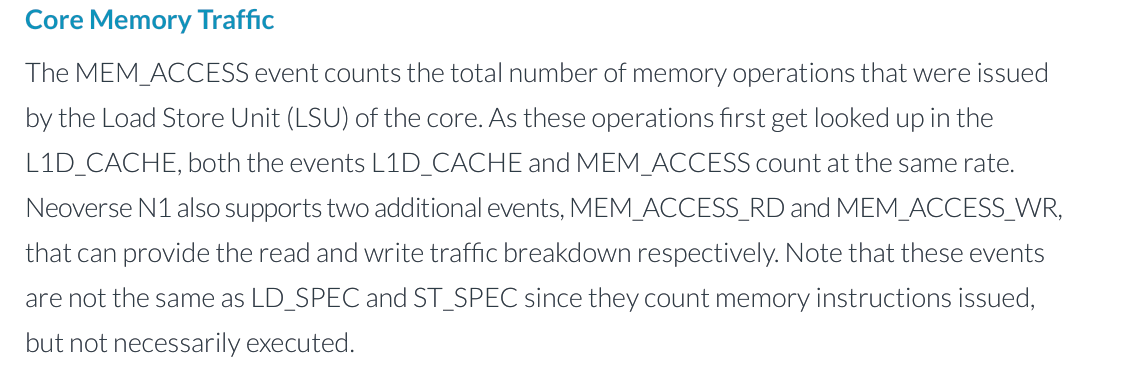


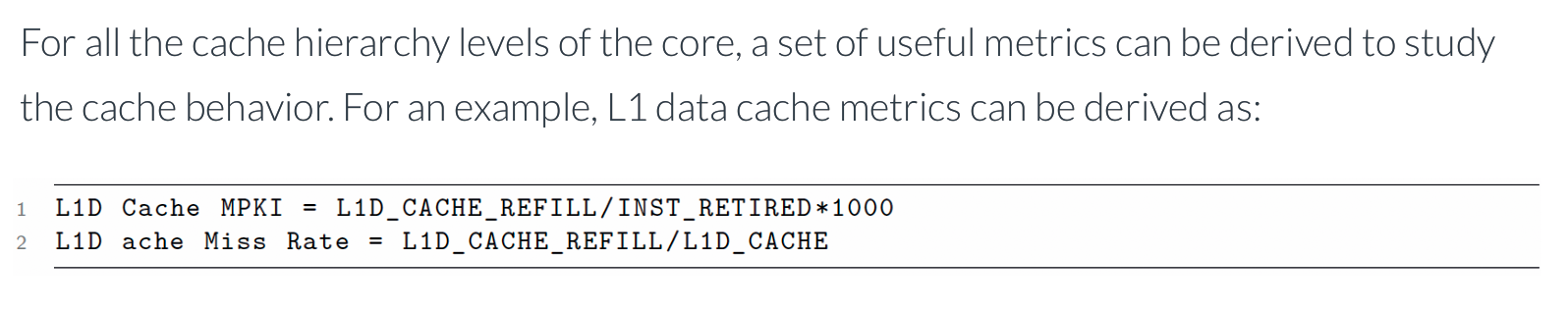


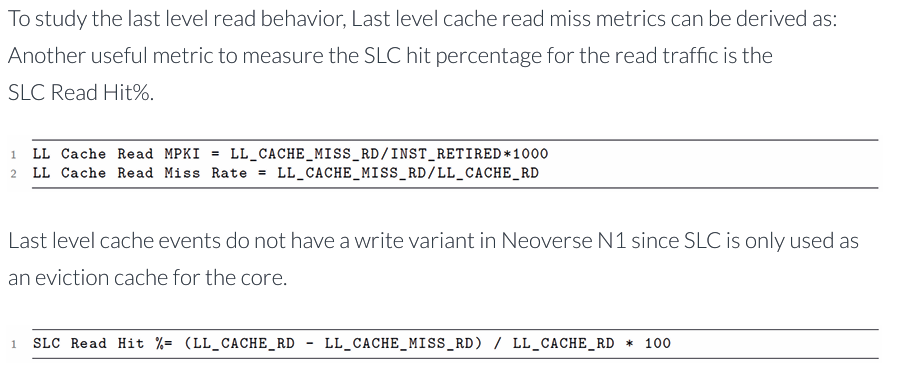












**Counting :**

* This event statistics help to characterize the overall workload execution behavior, without providing any details on where in the program a particular event occurred.
* Best approach for an initial workload characterization exercise to identify performance limitations of the workload.

**Event Based sampling:**

* This overflow interrupt records the event count and also the instruction pointer address and register information.
* Used to construct profiling information about the application, including stack trace and function level annotations.
* It is easy to locate the libraries and code portions that contribute to the large portion of the sampled event.
* Event Sampling mode is highly useful for hot spot analysis on a large portion of code, which relies on a statistical approach to sample different events over a large portion of time or code.
* Sampling delay, i.e., between the counter overflow and interrupt handler, which causes skid in the data obtained, that is, data stored during the sampling process and may not be the exact point where the event occurred.
* Another issue comes from the speculative execution style of the processor, where some instructions that executed and triggered events may not be valid if they were on the wrong code path.
* An easy test to verify that PMU events are being counted properly is to use the perf stat functionality of Linux perf tool to count instructions and cycles.

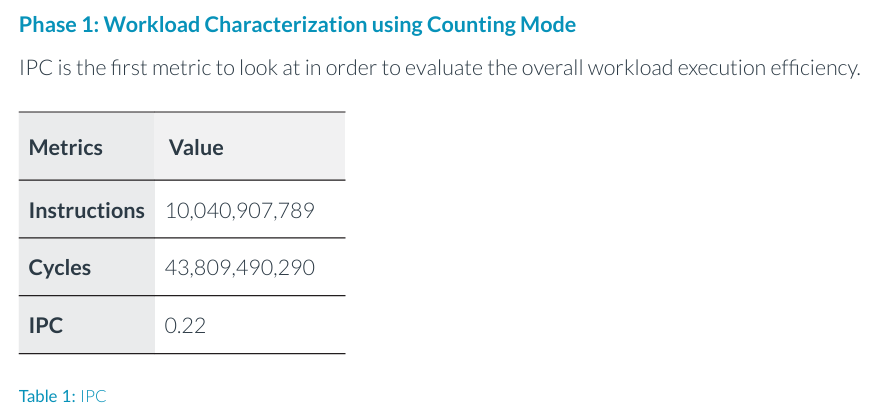
Use Case Study of CPU Performance Analysis :

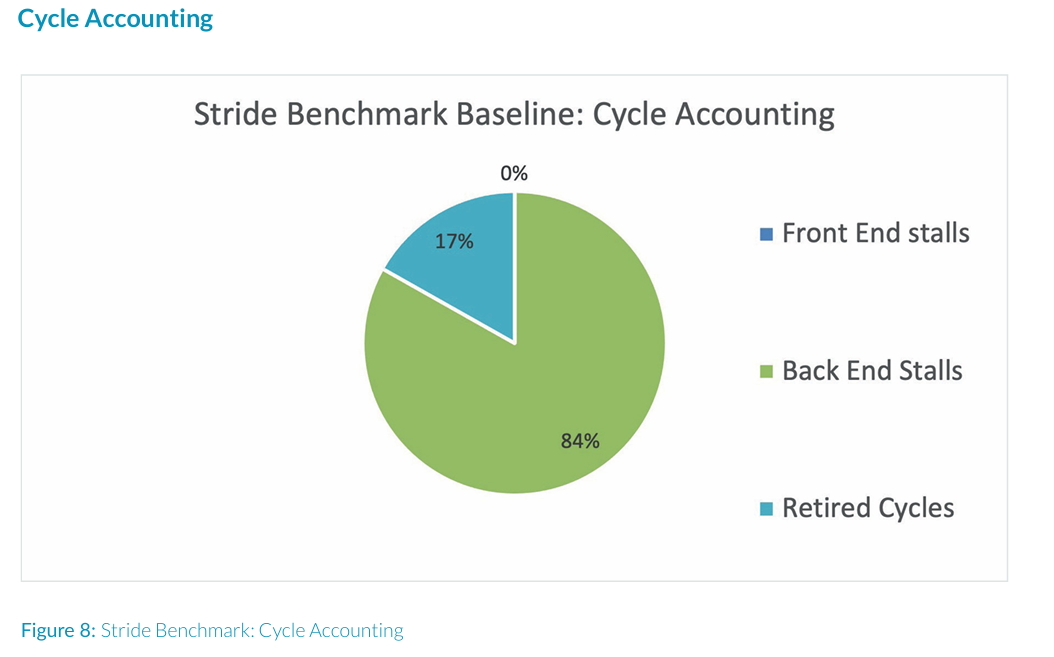
Use the performance analysis methodology outlined in section [Chapter 3] **for workload characterization and hot-spot analysis with core PMU metrics**captured from Neoverse N1 systems using Linux perf.

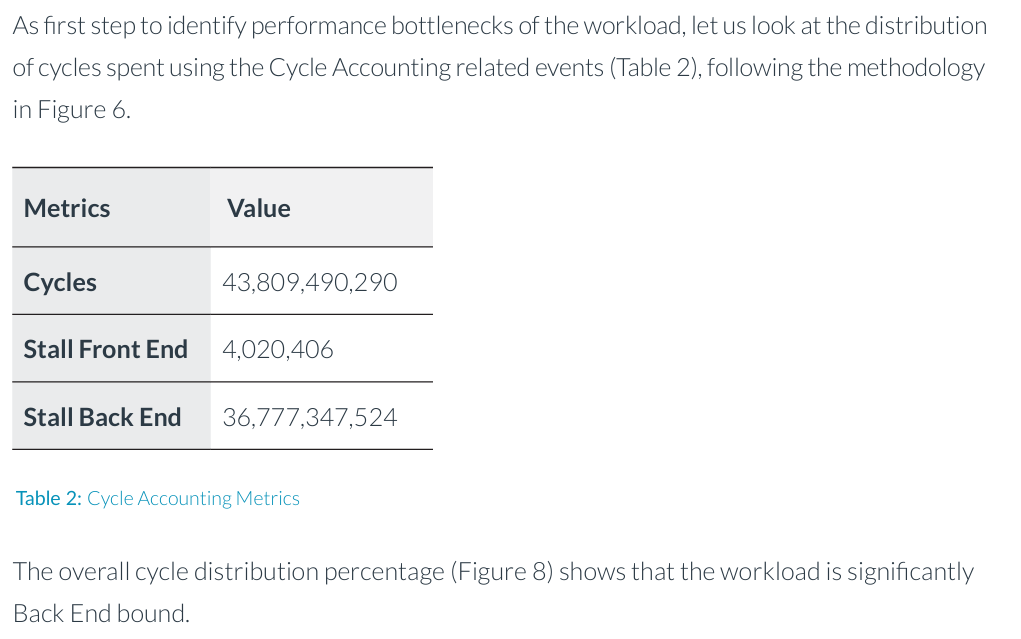
* Example workload characterization case study running on the Neoverse N1 Software Development Platform(N1SDP), which has 4 Neoverse N1 cores.
* The PMU events recommended [Chapter 2] are collected in batches of 6 events at a time using Linux Perf tool, “perf stat”.

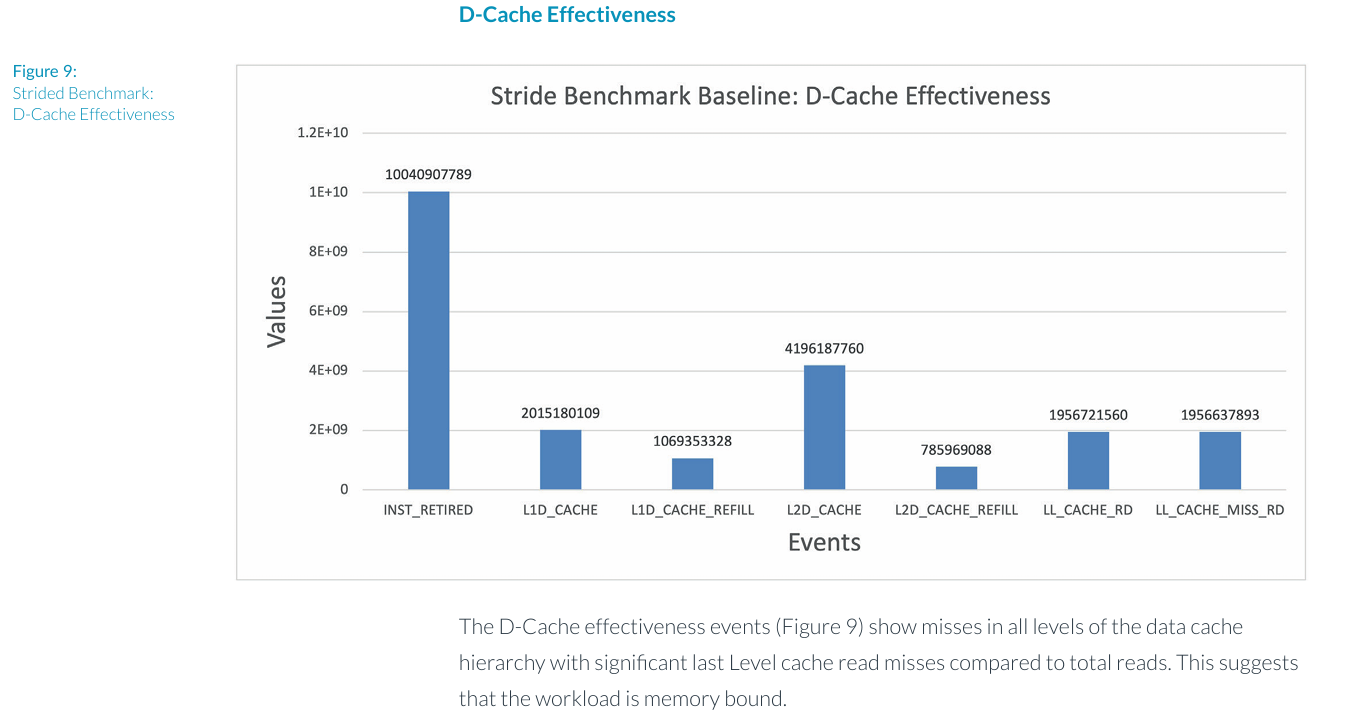
Stride Benchmark from the DynamoRIO tests :

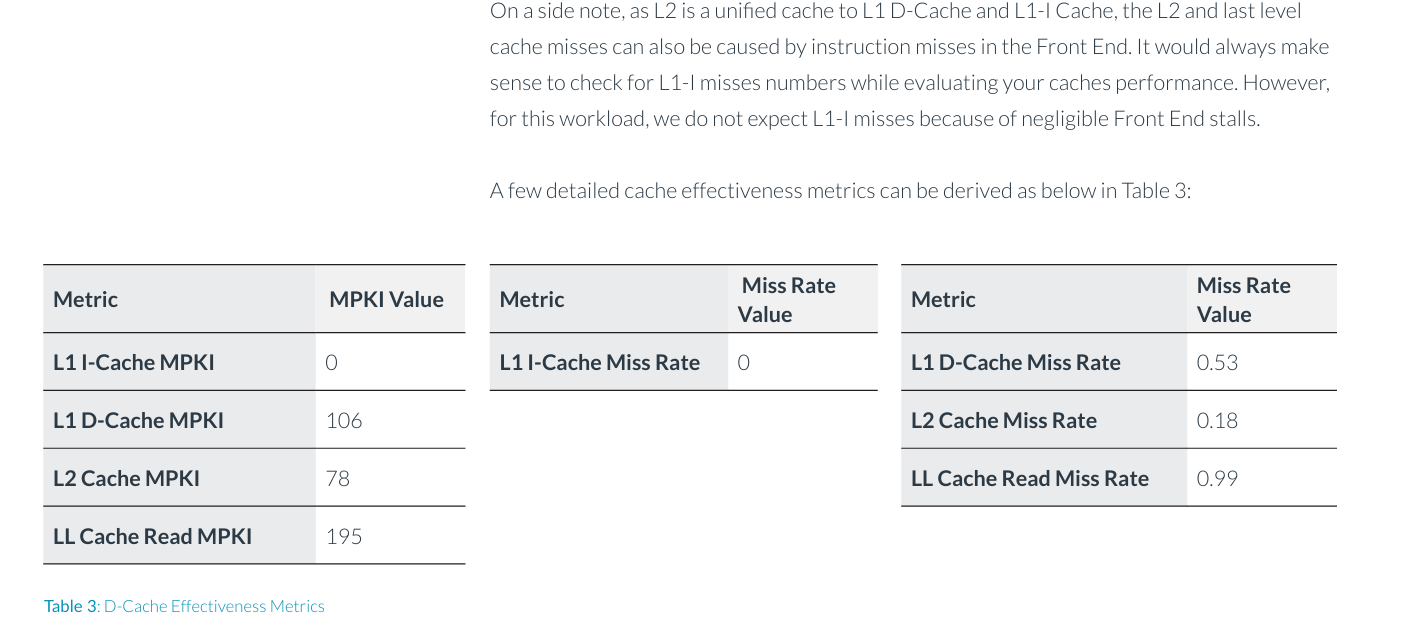
* The stride micro-benchmark 10 is a pointer chasing benchmark that **accesses values in a 16MB array**, with array position being determined by the pointer being chased.
* The pointer position is a**function of a constant value set in the array before the pointer chasing kernel runs.**
* The maximum achievable **IPC is 4**on Neoverse N1, which is typically **achieved by small & heavily optimized kernels** rather than large applications.

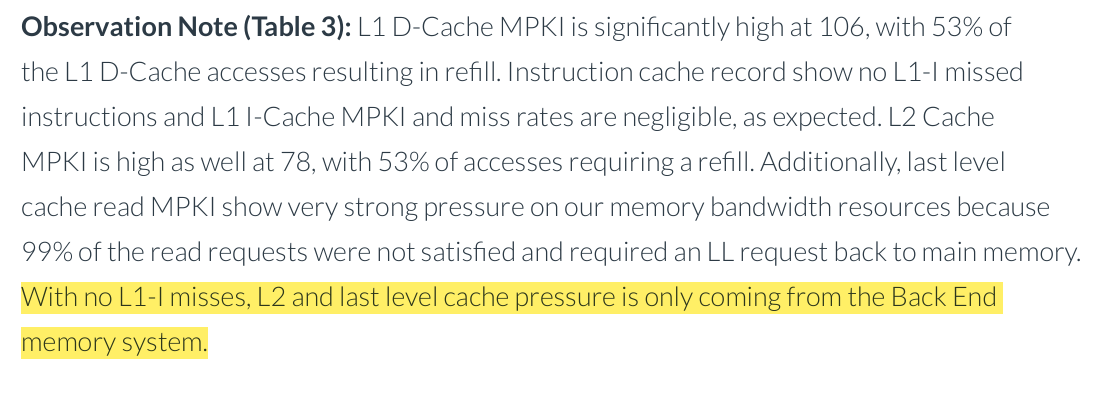


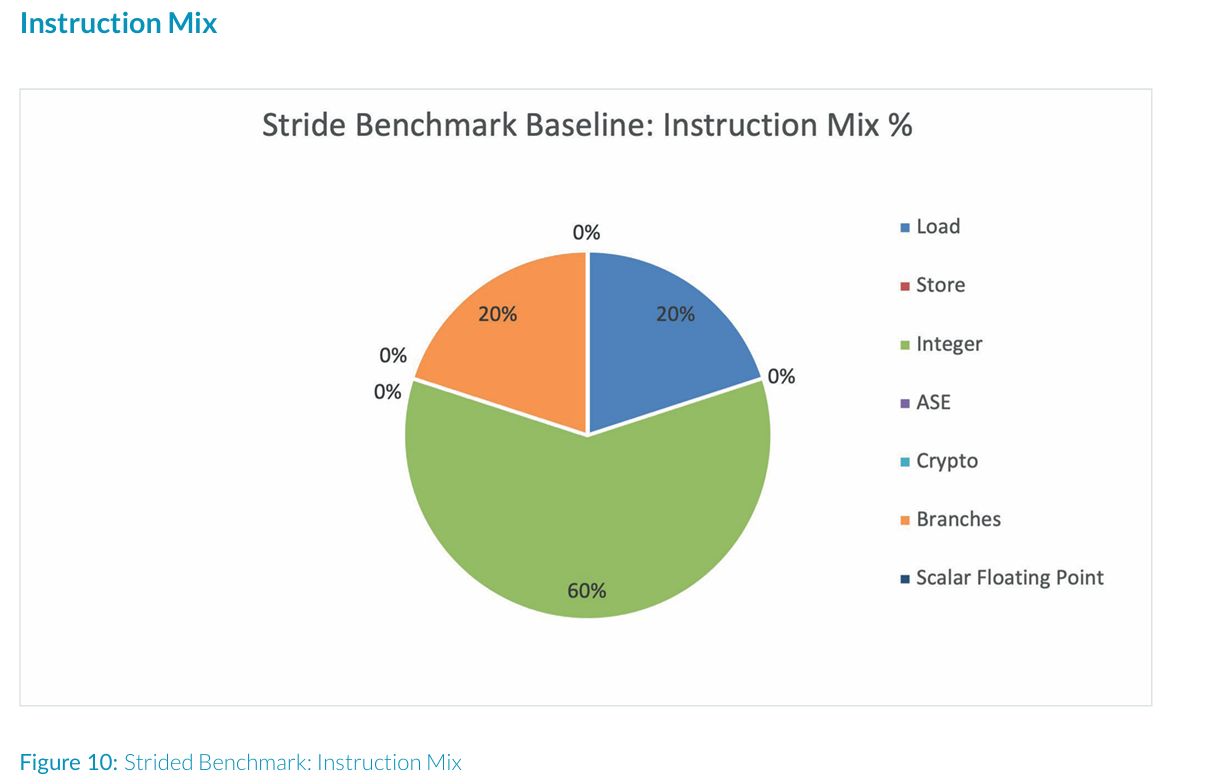


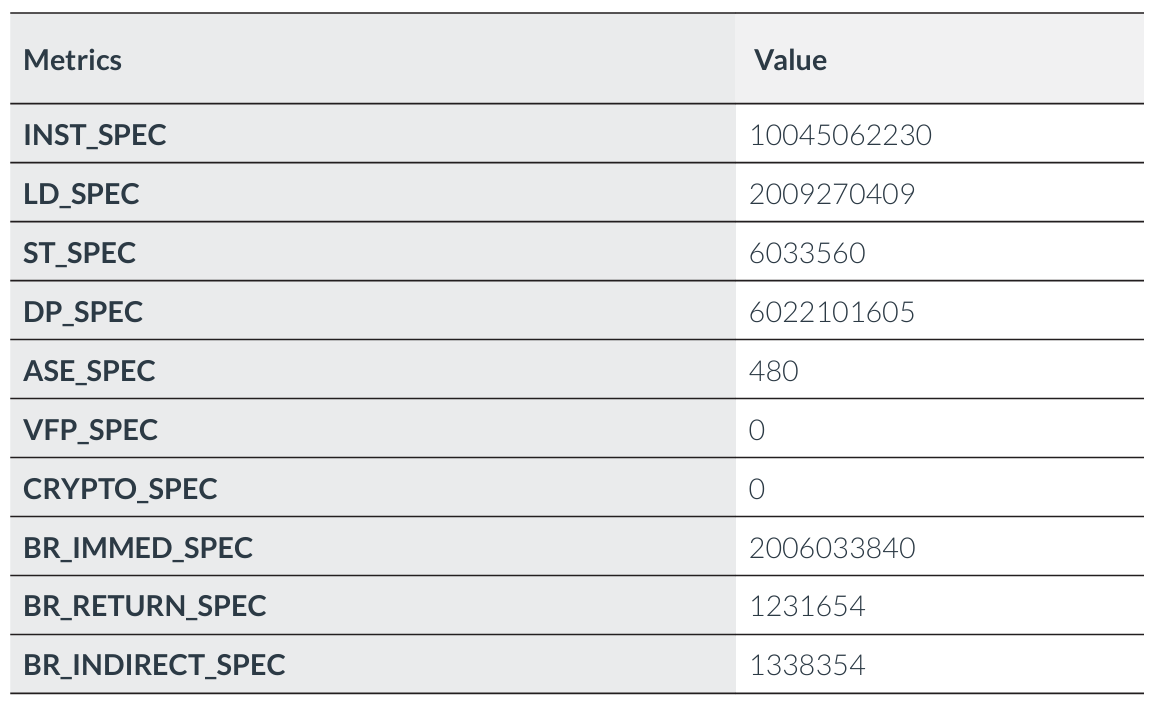


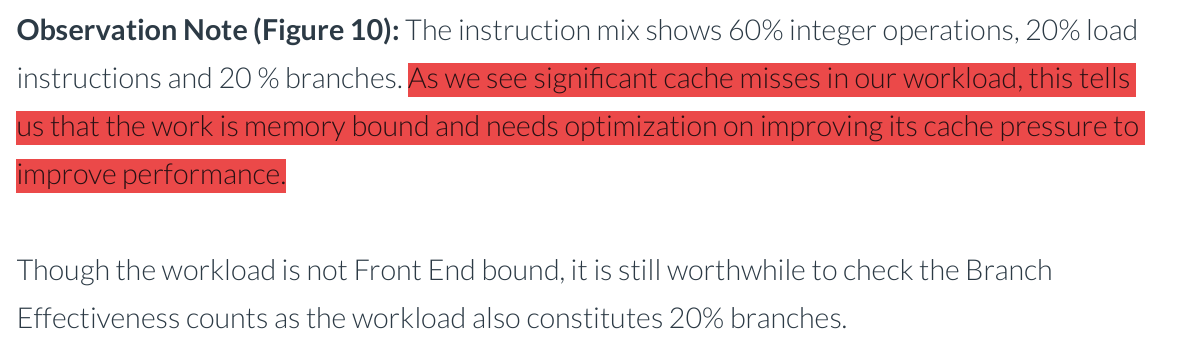


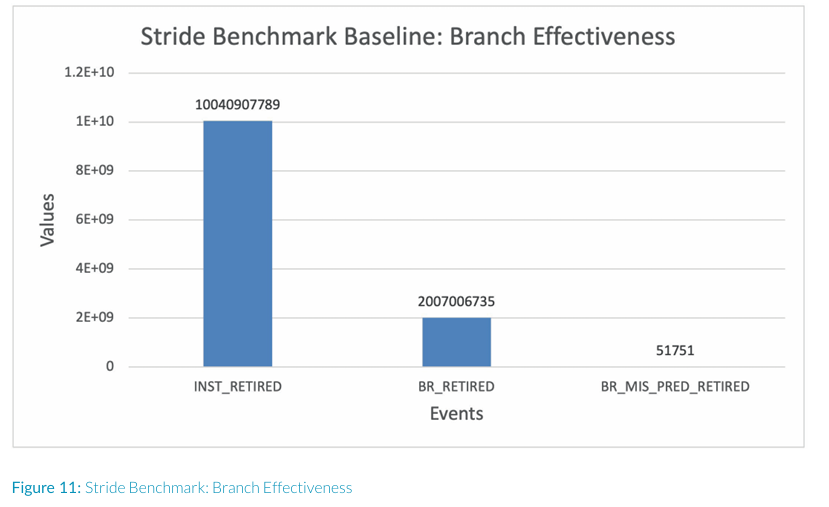


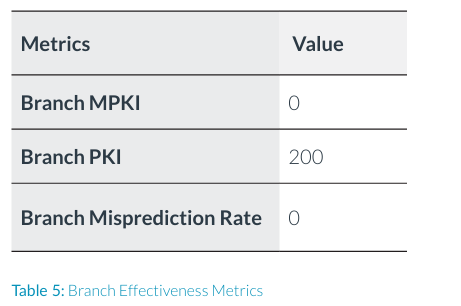


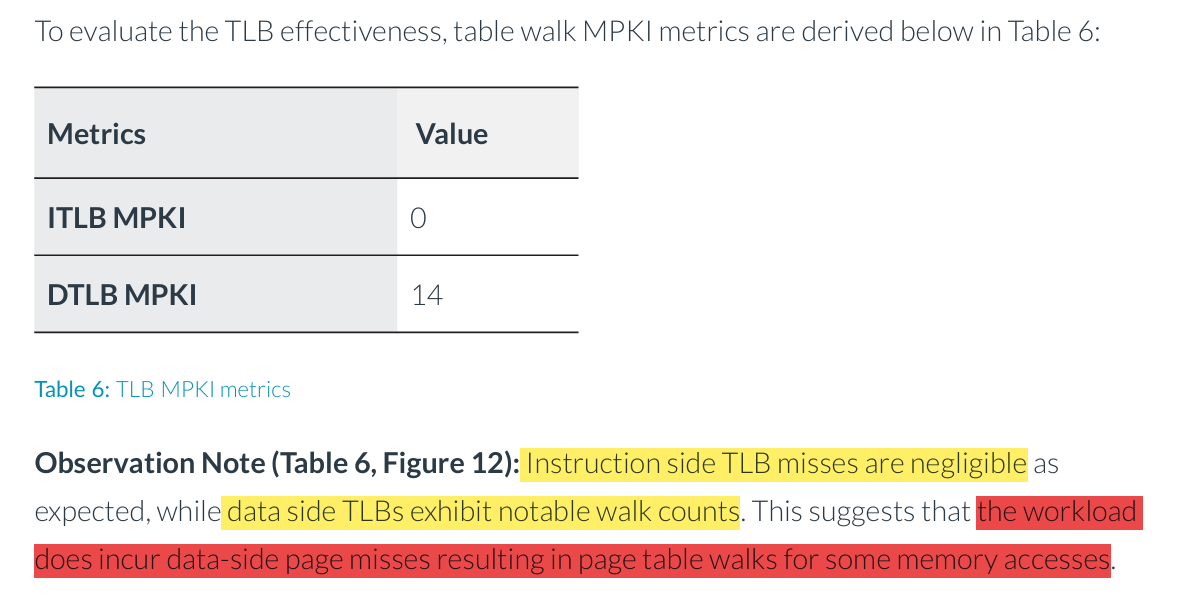


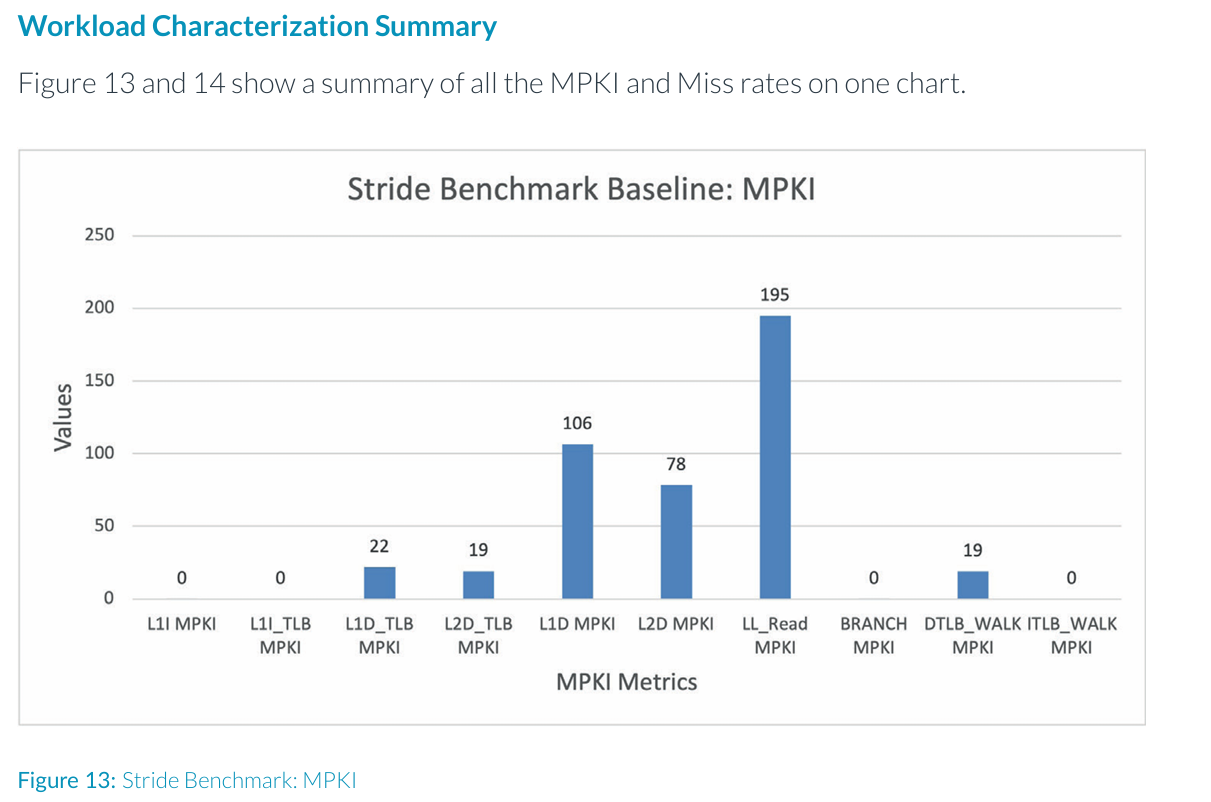


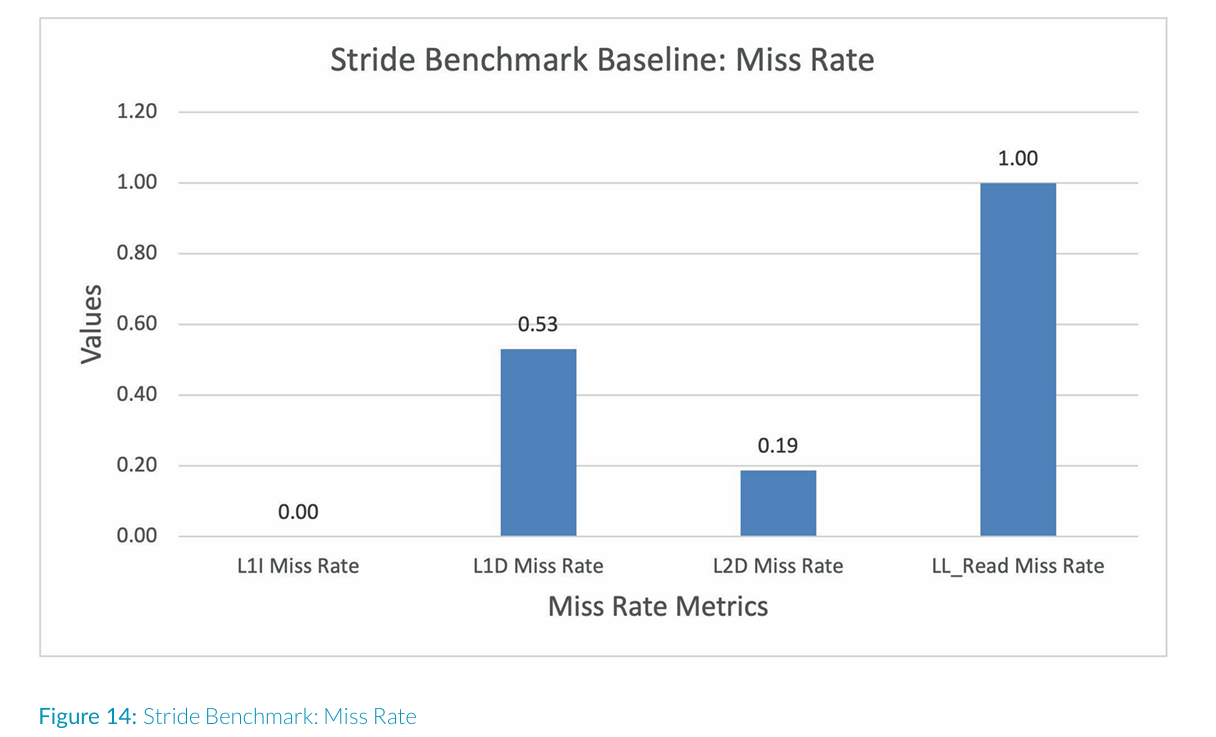












**Conclusion of  Use Case or Workload : Strided Benchmark**

* Poorly performing with**IPC = 0.22.**
* The workload is heavily Back End bound with a**very high Back End stall rate of 83%.**
* The workload has **60% integers**, **20% branches**and **20% load operations.**
* The workload exhibits **significant Back End pressure in the data cache side**, with L1D MPKI of 106, L2 MPKI of 78 and Last Level Cache Read MPKI of 195.
* The **Front End of the CPU is operating smoothly** with stats like Branch MPKI, L1I MPKI and ITLB MPKI being negligible, which corresponds to the zero Front End stalls.

**Conclusion of this Benchmark :**  The characterization evidence supports the workload behaviour that **there is a memory bottleneck in our system** and we should next investigate how to address it.

Understand the Bottlenecks :

* Back End stalls
* Hierarchical D – Cache Events

