Analysis on Coherent Memory Systems Within AI Training Workloads

Ziang Zhao[a]
Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1323 West Dayton Street, Madison, WI, U.S.A.

Keywords: Memory Consistency, Cache Coherence, Artificial Intelligence.

Abstract: Within larger groups of multiprocessors and heterogeneous computing clusters, the problem of data coherence has been of increasing concern. As the computational devices become increasingly complex, so do the methods that were being used to ensure the data integrity. With the recent rise in the training requirements of Artificial intelligence systems, the demand for larger coherent data systems has risen significantly. This paper aimed to provide an overview and analysis of different data coherence methods and analyze their potential performance within an AI training workload, and concluded that the Artificial intelligence (AI) training models would often require memory systems to be efficient over access of training parameters over extended periods of time and ensure its reliability across the extended training process. The paper would mostly contain a theoretical analysis of different coherence methods and would aim at providing an upper limit of the performance of these different methods. Further research regarding the physical implementation of such coherent systems might be still required.

1 INTRODUCTION

With the recent developments of Artificial intelligence (AI) systems, especially Large Language Models (LLMs), the computational platforms required for the training of these platforms often require a large number of processors. More specifically, most modern Large Language Models use large clusters of Graphical Processing Units (GPUs) for their training (Dubey et al., 2024). While the utilization of the large number of clusters allows a large amount of parallelism within the model to be exploited (Li et al., 2023), an important aspect during using these types of has been the importance of ensuring the data accessed could be consistent across multiple processing units. Two main concerns are often associated with consistent data access.

One of the concerns that is often considered is cache coherence. Cache coherence protocols are methods that ensure the individual caches can maintain a correct copy of the data, and are first used across multiprocessor systems with multiple Computational Processing Units (CPUs) (Sorin et al., 2022). Since its first establishment as part of the computing system, its usage has grown dramatically, both regarding on-chip communication between multiple connected processor clusters and network communication of clusters. While enforcing a cache coherence protocol is often useful in ensuring the accuracy of data accesses, additional time and resources are often required for the implementation of such protocols, which could more often than not lead to increase memory access time.

Another important consideration must be the consistency of data access for the specific memory system. Data consistency protocols ensure that all memory accesses towards a central memory system remain in a reasonable sequence, which would help to ensure that memory access results were reasonable and expected by the programmers. Even though most memory systems would often benefit from more intuitive sequential consistent models, a more relaxed form of memory consistency structure could more often than not accelerate relative memory accesses (Steinke & Nutt, 2004). However, the utilization of such alternate methods of memory consistency could often lead to requirements of additional support from the software, and sometimes an expected erroneous

[a] https://orcid.org/0009-0004-8047-4851

response within the system could require alternate methods of recovery.

While existing methods for maintaining cache coherence and data consistency could still be useful even as future requirements for the training of AI systems become increasingly complex, the need for innovative methods for ensuring consistency are still present. The following sections of the paper would first provide a brief introduction of different methods being used for ensuring memory consistency and coherence across history, and then provide an analysis of the newer requirements of such a memory model that has been brought forward through large-scale AI training. This paper would aim at providing a general summary of the current methodologies, and propose a direction for the potential aspects of a new memory consistency protocol.

2 RELATED WORKS

A coherent memory system has been established as one of the most important aspects of a memory system for an extended period of time ever since its first establishment. The following sections within this paper evaluate several cache coherence and consistency systems in chronological system as they were first established and provide a brief summary on its usages and potential problems.

2.1 Past coherence and consistency protocols

A cache coherence protocol is being first established as a method to ensure the consistency of memory access across a shared memory multiprocessor, where each processor cores often utilize its own near-core caches for data storage. The protocol would ensure that for each cache that stores the same data at the same memory address, either it stores the most recently updated value or an outdated value that would be invalidated on next access (Al-Waisi & Agyeman, 2017). Such protocol is often used in order to maintain a coherent memory system, which requires that all memory accesses issued by every processor is arrived in the same order as the processor and the memory system, and all memory read operations should return the same value as the last memory write (Grbic, 2003). For most earlier coherence protocols, they often adopt a snoop-based protocol, where each dedicated cache near the core would need to monitor for any memory accesses to ensure that all the caches would be able to store the most recent version of the memory. This kind of coherence protocols were extremely easy to implement thank to the shared-bus memory structure that enables the hardware implementation of such protocols. An example of such protocol would be the MSI protocol, where each shared memory cache block could be in one of the three states available, namely Modified (M), Shared (S) and Invalid (I), which is also where the protocol got its name. A cache line is being held in the Modified state if it is the only cache that contains the most up-to-date copy, is held in the Shared state if there are multiple caches containing the same most up-to-date copy, and Invalid state if the copy within the cache is outdated. Figure 1 shows the state transition diagram of the MSI protocol.

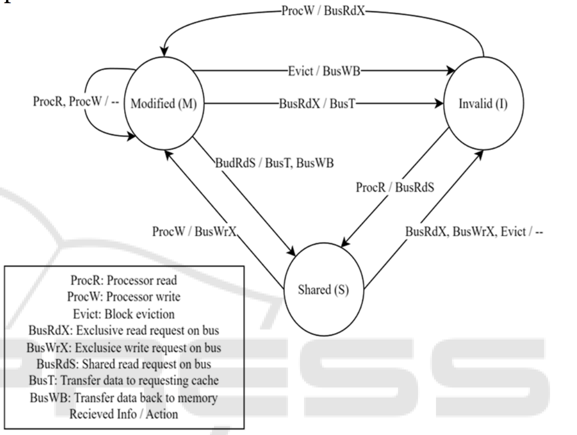
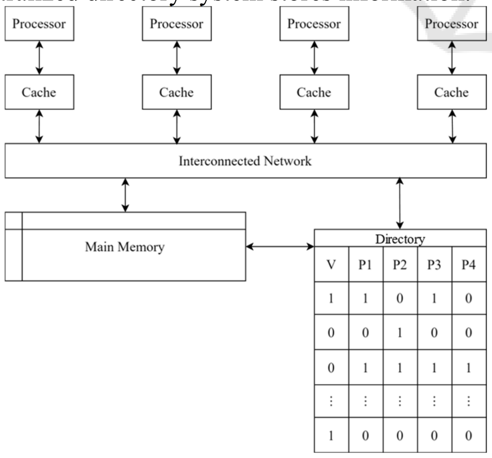


Figure 1: The state transition diagram of the MSI protocol, outlining the state changes and the bus activities. (Photo/Picture credit: Original).

In order to maintain the coherent memory system, additional constraints in terms of the consistency of the memory would be required to be maintained. The memory consistency model ensures that all of the memory accesses would follow a given rule from the eye of the processor and the programmer by extension. One of the earliest and the most common form of memory consistency models would be sequential consistency (SC), where the memory operations from all processors are executed in such an order that all operations from the same core occurs in the same sequence as both the memory system and the processor side (Zucker & Baer, 1992). The model has been of wide usage for most older systems since it has been the base assumption for most programmers, as it matches the expected performance of a centralized shared memory system within a multiprocessor setting.

2.2 Modern Memory Coherent Methods

As the number of available computational cores becomes increasingly complex, the general snooping-based cache coherence protocols ceased to become efficient. The problem mostly rise from two limitations, one physical and one theoretical. The physical limitation of placing multiple caches on the same communication bus would severely limit the ability of transfer of broadcasted information across multiple caches, and the requirement for one cache to constantly listen to other caches for potential cache change information led to a great decrease in the performance of caches. In order to better mitigate the bandwidth of the shared memory bus, most modern cache coherence protocols utilize a directory-based approach for multiple cores, while limiting snooping-based protocols within smaller cluster of processors. A directory is often considered to be a centralized location of storage for the relative information of a specific cache line across multiple different processors, and the utilization of these kinds of structures help to relieve the stress on the shared memory bus, as it is no longer required for each cache to continue to monitor for all memory accesses (Agarwal et al., 1988; Grbic, 2003). The directory would be able to work by using a map for each cache line within memory, where each bit of the map represents where the newest copy of the data for this cache line is being stored. Figure 2 shows how a centralized directory system stores information.



V	P1	P2	P3	P4
1	1	0	1	0
0	0	1	0	0
0	1	1	1	1
⋮	⋮	⋮	⋮	⋮
1	0	0	0	0

Figure 2: Outline of a centralized directory model, showing the different aspects of the memory system. (Photo/Picture credit: Original)

While a centralized directory coherence model would be able to mitigate the bandwidth requirement and bandwidth occupation for a snoopy based coherence models, some more modern systems decided to utilize a distributed directory model, where different cache coherence protocols are used in unison. A very common early way of implementing cache coherence has been the utilization of a snoopy based protocol within small CPU clusters, while a distributed directory model is being used for communication of coherence within different clusters (Chaiken et al., 1990; Shieh, 1998). Such constructions of the main memory are often considered to be a form of Non-Uniform Memory Access (NUMA), which is often considered to be an essential aspect of modern cache coherence design.

In terms of memory consistency, in order to increase the efficiency of memory accesses and to enable better support for multiprocessors, the based assumption of the sequential consistency of the memory systems is often relaxed, as this could allow the reordering of certain memory requests, which could result in an increase in the performance of the entire system. At the higher level, most contemporary memory models only require certain order of memory access to be preserved. For example, the SPARC ISA required that all processors to support Total Store Ordering, which would only ensure that all memory stores would be completed before the next load (The SPARC Architecture Manual, Version 8, 1992). Such relaxed models could allow the implementation of a store buffer, which would allow memory access requests to be processed more efficiently without sacrificing the correctness of the programs.

2.3 Consistent Memory in General Purpose Graphical Processing Units (GPGPUs)

As graphical processing units are becoming increasingly capable, more and more general purposed tasks were often dedicated towards these processors, given to the rise of general-purpose graphical processing units (GPGPUs). Such processors are most notable due to a larger number of weaker processing cores, which would be able to allow for a faster execution of parallel computational tasks, especially matrix operations, which were traditionally dedicated towards graphical rendering tasks, hence the name. As the field of machine learning and deep learning algorithms developed, which heavily utilized matrix operations during their training process, the need for ensuring a consistent memory within GPGPUs was rising. Most GPGPU do not utilize a hardware transparent consistency system, though it is very common for some GPUs to implement a form of temporal coherence system that ensures that the data would be available at the shared line between dedicated cores and across cores at times of requirement (Singh et al., 2013). While such processes would add additional overhead towards the operation of the codes, the performance gains outweigh the drawbacks of such systems.

3 ANALYSIS OF AI TRAINING APPLICATIONS

As AI training Models becoming increasingly larger since the deployment of large language models such as GPT-3, there has been an increasingly usage of parallel training models, where a variety of parallelisms were being utilized within such models (Li et al., 2023). The training data is often distributed across multiple GPUs, with the training task itself also being distributed across the GPUs with the tasks often pipelined. During the actual training of models, the training weights would be required to be accessed multiple times across the training process, while the training data would often only require more sporadic accessed. Due to a large amount of data and larger number of processing units, memory reliability has also become an important aspect of consideration within modern systems (Dubey et al., 2024). Therefore, an outline of the memory consistency requirements for AI workloads would include a high support on sporadic temporal locality of the data, higher storage capacity for accessing of consecutive weights, and a highly reliable memory to prevent potential access errors due to the larger memory network.

3.1 Efficiency Considerations

Given the parallel nature of the GPU, the efficiency of the memory system during training models would be essential. While the GPU would often distribute the data across multiple cores for its calculations, it would often require to access multiple training parameters as the models are becoming increasingly complex (Dubey et al., 2024; Li et al., 2023). Therefore, the special locality of the memory workload, given that it has already been exploited by the parallel structure of the GPU, would be of lesser importance to be exploited by the processors. The more important part would be ensuring that the temporal locality could be exploited in a better method, probably utilizing a larger cache structure and finding a balance between caching invalidation and the utilization of quick memory accesses.

3.2 Reliability Considerations

Since the models were often training for extended periods of time as a large system, it would be very likely for memory accesses to fail during its training process (Dubey et al., 2024). Therefore, the actual memory consistency model should also be performing consistently through an extended periods of time and would require less time for recovery. This would often be able to be satisfied through the introduction of error correction codes within the memory system and a constant self-reporting of faulty occurrences of memory access failures. A more axiomatic and well-ordered memory consistent system could be also utilized to ensure the correctness.

4 DISCUSSIONS

While this paper touches a number of different memory consistency techniques, including cache coherence and memory consistency protocols, this paper would only serve as a theoretical analysis of the potential impacts and requirements of the memory system of artificial intelligence training systems. Further work regarding the physical reality of implementing such systems on a larger scale and a more rigorous mathematical analysis would be likely required for the realization of such novel systems. Despite these concerns, this paper would be able to leave a good foundation for future works to be built upon and help to inspire further research against these newer models.

5 CONCLUSIONS

This paper briefly outlines the evolution of coherent memory models, specifically cache coherence and memory consistency protocols. This paper then goes on to summarize the requirements that would be required within an AI training workload, which would be essential. In summary, memory consistent systems have come a long way since their first introduction with theoretical models of multi-core systems. While these models surely evolved a lot in terms of their efficiency along the way, the newer applications, specifically these related to artificial intelligence, would more often than not require newer design philosophies that were less performance focused and more reliability focused. Future memory models, therefore, should utilize an applicationforward approach regarding their designs to better enable utilization of their efficiencies and structures.

REFERENCES

Agarwal, A., Simoni, R., Hennessy, J., & Horowitz, M. 1988. An evaluation of directory schemes for cache coherence. ACM SIGARCH Computer Architecture News, 16(2), 280–298.

Al-Waisi, Z., & Agyeman, M. O. 2017. An overview of on-chip cache coherence protocols. 2017 Intelligent Systems Conference (IntelliSys), 304–309.

Chaiken, D., Fields, C., Kurihara, K., & Agarwal, A. 1990. Directory-based cache coherence in large-scale multiprocessors. Computer, 23(6), 49–58.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., & Fan, A. 2024. The Llama 3 Herd of Models. arXiv Preprint arXiv:2407.21783.

Grbic, A. 2003. Assessment of cache coherence protocols in shared -memory multiprocessors [Ph.D., University of Toronto (Canada)]. In ProQuest Dissertations and Theses (305258117). ProQuest Dissertations & Theses Global.

Li, S., Liu, H., Bian, Z., Fang, J., Huang, H., Liu, Y., Wang, B., & You, Y. 2023. Colossal-ai: A unified deep learning system for large-scale parallel training. Proceedings of the 52nd International Conference on Parallel Processing, 766–775.

Shieh, K.-Y. G. 1998. A hybrid directory-based cache coherence protocol for large-scale shared-memory multiprocessors and its performance evaluation [D.Sc., The George Washington University]. In ProQuest Dissertations and Theses (304443768). ProQuest Dissertations & Theses Global. https://ezproxy.library.wisc.edu/login?url=https://www.proquest.com/dissertations-theses/hybrid-directory-based-cache-coherence-protocol/docview/304443768/se-2?accountid=465

Singh, I., Shriraman, A., Fung, W. W., O'Connor, M., & Aamodt, T. M. 2013. Cache coherence for GPU architectures. 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), 578–590.

Sorin, D., Hill, M., & Wood, D. 2022. A primer on memory consistency and cache coherence. Springer Nature.

Steinke, R. C., & Nutt, G. J. 2004. A unified theory of shared memory consistency. Journal of the ACM (JACM), 51(5), 800–849.

The SPARC Architecture Manual, Version 8 (Version SAV080SI9308). 1992. https://sparc.org/technical-documents/

Zucker, R. N., & Baer, J.-L. 1992. A performance study of memory consistency models. ACM SIGARCH Computer Architecture News, 20(2), 2–12.