Āpta Appendix: Coherence Protocol Specification and Verification

1. Summary

In this appendix to the main paper [3], we describe how we model and verify the Āpta protocol. We also provide the complete specification of the directory controller i.e., object tracking controller (OTC), including transient states and actions, as a sample specification.

2. Assumptions

- A rack scale PCIe network with compute servers and memory servers equipped with micro-controllers to employ CXL.mem protocol over the PCIe physical network [5, 2, 1, 4].
- This coherence protocol operates inter-server i.e., between the last-level cache (LLC) on the compute servers and OTC on the memory server. The request and response transactions in the protocol are issued at the granularity of objects. Recall, objects are variable-sized, consecutive blocks of memory represented by a triplet (`objID`, base PA, size).
- The OTC directory holds entries are at the granularity of objects and is inclusive of compute server LLCs in the system i.e., it holds directory entries for a superset of all the objects cached in all the compute server LLCs.
- The protocol uses a strict hierarchical communication pattern. This means the LLC, for a request associated with disaggregated memory, communicates only with the OTC and never communicates directly with any other LLCs. On receipt of a request, the OTC issues a request to other LLCs if required. Likewise, responses from the OTC are sent to the LLC, which then forwards it to other lower level L1 or L2 caches, as required.
- The LLC silently evicts any of the disaggregated memory cache lines which are clean (in shared state) without issuing any coherence requests (i.e., PutS) to the OTC directory. If the LLC receives an invalidation request for objects that is not present in the LLC, the invalidation is immediately acknowledged with no other action.
- The protocol assumes that FaaS applications are written to be race-free, i.e., no `put` or `get` can occur during an ongoing `put`. If races do occur, the protocol stalls or rejects such requests until the ongoing `put` request completes.

3. Verification

We have verified the complete specifications of the allow-based and the deny-based protocols using the Murϕ model checker for safety and deadlock-freedom. A protocol is deemed to be safe if it enforces the Single-Writer-Multiple-Reader invariant (at any time there exists a single writer or multiple readers) and the data-value-invariant (a read always returns the value of the most recent write) across all of the caches, on all compute servers. Enforcing these invariants implicitly ensures strong consistency of caches and memory.

We have verified the protocol for the reference system modelled in our evaluation i.e., 3 compute server, each with a single shared LLC and 1 memory server with the directory (OTC).

4. OTC Directory Spec

The complete specification of the OTC directory with the request events/messages, state transitions and responses to the LLC is given in Table 1. It uses 2 stable states - Shared (S), Invalid (I) and 2 transient states - S^A, SI.

The states S and I mean the same as in the CXL protocol. The state S^A signifies that the new version of the object is cached in Shared state in one or more compute servers and there are pending invalidation-acknowledgments from one or more compute servers for the old version of the object. The state SI indicates that the entry from the directory cache set is being evicted. While in the SI state, any requests to the object while in this state, are stalled until all invalidations are acknowledged and an entry in the directory becomes available.

The table defines transitions for 3 message types received from the LLC - `Get`, `Put` and `Inv_Ack` and 1 `Replacement` event. The replacement event triggered on the eviction of an entry from the directory cache set. The `Put` message is triggered on the completion of an object write request. The `Get` message is triggered at the beginning of an object read request. The `Inv_Ack` message is triggered when a LLC responds to an invalidation with an acknowledgement message.

5. Notation

- Receive message: ?Message
- Send message: !Message(Destination, *Payload)
  *Payload is optional
- Next state transition: → Next State

- If a message arrives in a state where the table entry is grey it is considered to be stalled.
- Write to memory : MemWr
- Read from memory: MemRd
- PUT Controller: PUTC
- GET Controller: GETC
- Object Tracker sub-controller: OTC

References

[1] D. Gouk, M. Kwon, H. Bae, S. Lee, and M. Jung, "Memory pooling with cxl," *IEEE Micro*, vol. 43, no. 2, pp. 48–57, 2023.

[2] M. Ha, J. Ryu, J. Choi, K. Ko, S. Kim, S. Hyun, D. Moon, B. Koh, H. Lee, M. Kim, H. Kim, and K. Park, "Dynamic capacity service for improving cxl pooled memory efficiency," *IEEE Micro*, vol. 43, no. 2, pp. 39–47, 2023.

[3] A. Patil, V. Nagarajan, N. Nikoleris, and N. Oswald, "Apta: Fault-tolerant object-granular cxl disaggregated memory for accelerating faas," in *Proceedings of the IEEE/IFIP 53rd Annual International Conference on Dependable Systems and Networks*, ser. DSN '23, 2023.

[4] D. D. Sharma, "Compute express link: An open industry-standard interconnect enabling heterogeneous data-centric computing," in *2022 IEEE Symposium on High-Performance Interconnects (HOTI)*, 2022, pp. 5–12.

[5] ——, "Novel composable and scaleout architectures using compute express link," *IEEE Micro*, vol. 43, no. 2, pp. 9–19, 2023.

Table 1: Object Tracker sub-controller: OTC requests and responses

State	Replacement	?Get	?Put	?Inv_Ack
I		MemRd add Sharer !Get_Ack(GETC, Data) $\rightarrow$ S	MemWr add Sharer !Put_Ack(PUTC) $\rightarrow$ S	
S	!MCAST(Inv, sharers) inv_cnt = sharers.count clear sharers $\rightarrow$ SI	MemRd add Sharer !Get_Ack(GETC, Data) $\rightarrow$ S	MemWr !MCAST(Inv, sharers) inv_cnt = sharers.count clear sharers add Sharer !Put_Ack(PUTC) $\rightarrow S^A$	

S^A		MemRd
		add Sharer
		!Get_Ack(GETC, Data)
		$\rightarrow$ S
				inv_cnt --
				if inv_cnt == 0 {
				$\rightarrow$ S
				}

SI				inv_cnt --
				if inv_cnt == 0 {
				$\rightarrow$ I
				}