MRSW Coherence & Consistency Protocols

	Sequential	TSO	PSO	Release	Acquire	Scope
Home; Invalidate	[11,12,13]			[1,2]	[9]	[5]
Home; Update						
Float Home; Invalidate				[2]		
Float Home; Update						
Directory; Invalidate	[3]					
Directory; Update						
Dist. Dir.; Invalidate	[6]		[4]	[7]	[7,8]	
Dist. Dir.; Update				[7]		

Release consistency -- i.e., consistency between nodes are eagerly maintained (via invalidation or update).
Acquire consistency -- i.e., consistency between nodes are lazily maintained.

In a home-based coherence protocol, the "home" node is guaranteed coherency with respect to the memory model i.e., the home node is always guaranteed to have the most recent rendition of data in its storage. Inversely, whoever have the most recent rendition of data in its storage can become the home node, which is the case for "floating home" protocols.

In floating home protocols, some orchestration is needed for a stale node to be able to have its request message routed to the current home node. A simple implementation is to provide an orchestrator node which knows the order in which each node were "home" -- some garbage collection required.

A scope-consistency protocol is one where each sharing is associated with a scope -- only loads and stores within the scope's participants is guaranteed to respect the memory model.

Many newer systems don't even expose writable data-to-compute (e.g., Grappa) -- to write, you

have to move compute towards data. Reading is no-problem though.

[7]'s case for acquire-invalidate holds only for MRSW (I think?).

[10] writes an Itanium hypervisor to respect atomic and fence instructions. It's interesting, but because it emulates NUMA no coherence is applied between nodes -- there simply is no cache for remote nodes.

[11] should be familiar. Home node also acts as the lock manager node (for simplicity).

Home seems intuitive for implementing such a system, e.g., [12], which subconsciously used home-based DSM to implement their hypervisor mm. Their interests were in hypervisors, though.

[14] is the Argo paper, which were compared with in [2]. Its coherence is entirely based on API function calls, which (ofc.) improves performance. Home-based.

References

[1] Shan, Tsai, and Zhang, "Distributed shared persistent memory", 10.1145/3127479.3128610
[2] Endo, Sato, and Taura, "MENPS: A Decentralized Distributed Shared Memory Exploiting RDMA", 10.1109/IPDRM51949.2020.00006
[3] Wang et al., "Concordia: Distributed shared memory with {In-Network} cache coherence", fast'21
[4] Cai et al., "Efficient distributed memory management with RDMA and caching", 10.14778/3236187.3236209
[5] Hu et al., "JIAJIA: A Software DSM System Based on a New Cache Coherence Protocol", 10.1007/BFb0100607
[6] Chaiken et al., "LimitLESS Directories: A Scalable Cache Coherence Scheme", 10.1145/106975.106995
[7] Carter, Bennett, and Zwaenepoel, "Implementation and Performance of Munin", 10.1145/121133.121159
[8] Keleher et al., "Treadmarks: Distributed Shared Memory on Standard Workstations and Operating Systems", EPFL
[9] Holsapple., "DSM64: A Distributed Shared Memory System in User-space", CalPoly
[10] Chapman and Heiser, "vNUMA: A Virtual Shared-Memory Multiprocessor", usenix'09
[11] Kim et al., "DeX: Scaling Applications Beyond Machine Boundaries", 10.1109/

ICDCS47774.2020.00021
[12] Zhang et al., "GiantVM: a type-II hypervisor implementing many-to-one virtualization", 10.1145/3381052.3381324
[13] Ding, "vDSM: Distributed Shared Memory in Virtualized Environments", 10.1109/ICSESS.2018.8663720
[14] Kaxiras, "Turning Centralized Coherence and Distributed Critical-Section Execution on their Head: A New Approach for Scalable Distributed Shared Memory", 10.1145/2749246.2749250