Juan-Carlos Cano, Ana Pont, Julio Sahuquillo,
and José-Antonio Gil
Polytechnic University of Valencia, Spain

The authors discuss the similarities in caching between the extensively studied distributed shared memory systems and the emerging proxy systems. They believe several techniques used in distributed shared memory systems can be adapted and applied to proxy systems.

The Differences between Distributed Shared Memory Caching and Proxy Caching

Computer architects introduced the concept of cache memory to reduce the gap between the large, slow, main memory, and the processor speed.¹ Since then, the concept of caching has spread to many fields of computer engineering, and a large amount of research has emerged for reducing latencies and improving

performances.²⁻⁵ This article focuses on caching in two widely used types of systems—distributed shared memory and proxy server systems—and stresses the main similarities and differences between these systems with regard to caching issues. This identifies possible caching mechanisms that we can improve by studying corresponding mechanisms on other types of systems.

Terms

DSM caching has been an active research area since the early 1980s and proxy caching since the early 1990s.⁶ In spite of the similarities between DSM and proxy caching, there are several remarkable differences that we can study from the point of view of the three major system layers: architecture, organization, and design.

In this article, the term *architecture* refers to the system's functional structure, visible to the assembly language programmer or the compiler. The term *organiza-*

tion refers to how we can structure the global system to implement the characteristics the architecture imposes. Organization is transparent to the programmer, and there are many organizations for a given architecture. Finally, the term *design* is a concept closely related to implementation and technology; it refers to the internal structure of a given system resource. There are several designs in each organization.

Criteria

The criteria we will describe are not fully orthogonal because some criteria are related to others. We labeled each criterion according to the system layer to which it belongs: architecture (A), organization (O), or design (D). Criteria identifying properties of the application that use the system are labeled as property.

GRANULARITY (O)

Both DSM and proxy systems must carry out actions to maintain the shared

data's coherence. We refer to the basic data-item size used in these actions as *granularity*, and the granularity level can vary a great deal among these systems. In DSM systems, the granularity level can be 1 byte, a word, a page, or a data structure. Granularity sizes in DSM systems mainly depend on their implementation. Hardware DSM solutions typically support smaller grain size (word or cache line), and software DSM solutions support coarse grain sizes (page or complex data structures). In proxy cache systems, granularity is variable in size—it ranges from bytes to megabytes.

REMOVAL POLICY (D)

In an ideal system (DSM or proxy), all the referenced data would fill the cache. Unfortunately, in a real system, the cache capacity is not large enough to place all the referenced data, so we need removal policies. These policies exploit data localities to improve performances.

In DSM, the classic Least Recently Used policy is a common approach because of its performance and easy implementation. In proxy cache systems, we take into account many other parameters—such as the frequency of use, download time, and document size—for placement and replacement policies. Thus, we can obtain more complex policies such as the LRU-MIN and LRU-HOLD.⁷ Understanding these removal proxy mechanisms might help us improve DSM cache policies.

SHARING MODE (A)

In DSM, we can classify the sharing or write mode according to the number of copies of the same data allowed in different nodes. Classification approaches are

- single reader/single writer (SRSW),
- multiple reader/single writer (MRSW), and
- multiple reader/multiple writer (MRMW).

Proxy systems use the MRSW approach, which is commonly used in DSM systems; however, newer DSMs increasingly use the MRMW approach.⁸ Because the coherence maintenance

solution strongly depends on the number of writers, coherence maintenance mechanisms in the newer DSM systems are more complex and expensive to implement than in proxy systems.

CONSISTENCY (A)

Consistency refers to a specific kind of memory coherence, and we can define it with the memory operation sequence observed by each system processor or client. In DSM, the solutions vary from relaxed to strict models. Strict consistency models typically increase the number of coherence operations and consequently the network traffic.

The proxy consistency models⁹ use the MRSW approach, so the related mechanisms are simpler than in DSM.

In both systems, we can classify consistency models as producer-initiated (PI) or consumer-initiated (CI),^{10,11} depending on the processor that initiates the processing activities and data transactions.

COHERENCE INITIATIVE (D)

Coherence policy determines how to keep all the shared data copies up to date. Two classic approaches are write-update and write-invalidate. Both are strongly linked with the system granularity; thus, the update policy is often used in fine-grain systems and write-invalidate in coarse-grain systems.

In DSM systems, invalidate-based coherence and prefetching mechanisms are typically based on CI approaches; data forwarding, update-based coherence, and locks are typically PI approaches.¹¹ In proxy cache systems, validation checks, proxy invalidate, and time-to-live are CI approaches; update and callback are PI approaches. All these techniques help determine whether files should be cached, removed, or validated.¹²

LATENCY BANDWIDTH (D)

Bandwidth refers to the maximum rate the interconnection network can propagate information once the message gets on the network. Sometimes, we use the term *throughput* to mean the network bandwidth delivered to an application. The time that a message spends in the

interconnection network, without including the related overhead, is *latency*.

A DSM system has a number of highly tuned computer nodes connected by a scalable interconnection network that must offer short latency and high bandwidth. Although latency and bandwidth are mainly technology dependent, we can use latency-tolerating techniques such as prefetching schemes and relaxed memory consistency models.

The perceived latency in proxy cache systems comes from both the server and the network. Although we can easily solve server latency by using faster computers or disk systems or a larger memory size, network latency is difficult to maintain under limits. This is because it depends on several parameters such as bandwidth, distance, network congestion, traffic characteristics, and roundtrip times.

Trying to scale the bandwidth to keep up with client demand is an expensive strategy in wide-area networks. Therefore, we can use prefetching techniques to improve proxy latency. Although prefetching techniques can improve the total latency the client observes, they can also dramatically increase the total bandwidth consumed—which happens with DSM. So, we must base these techniques on performance studies and use them carefully.

LOCALITY TYPES (PROPERTY)

To reduce the domain gap between processor and memory technology, the concept of cache memory has emerged. We can exploit the locality property present in data and instructions to achieve the cache's efficiency. Locality types are spatial and temporal. *Spatial locality* means that data or instructions to be used are probably "near" the running instruction or data being used. In traditional systems, spatial locality is exploited using larger cache blocks and prefetching mechanisms. *Temporal locality* means that there is a high probability that data or instructions will be reused in the near future. This locality is traditionally exploited by using cache hierarchy and removal policies that keep the most recently used data or instructions in the cache.

In proxy cache, spatial locality is likely to appear because the client usually maintains the first level integrated into the browser.

In DSM systems, in addition to these locality types, additional locality types such as the processor locality or access pattern locality exist as well.

In proxy cache, geographical locality is similar to processor locality using parallel applications in DSM. This implies that an object accessed by a client is likely to be accessed by neighboring clients in a relatively short time. The server can use global information, such as network topology and access patterns, to distribute its objects over other geographically distributed servers to improve performances.

SYSTEM POSITION (O)

Unlike DSM, we use caching in two ways on the Web: the client and the proxy caches. The former stores documents requested by the user, and the latter stores those requested by a community. In this way, the proxy cache behaves like a second-level cache; thus, after the initial client loading time, the number of requests tends to decline.

Another underlying point to consider is that we can hierarchically configure proxy cache systems so that supporting parent-child and sibling relations connected to a high-speed network can reduce bandwidth demand and the load on Internet server sites. In comparison, DSM systems are not hierarchically organized.

BEARING IN MIND the similarities between DSM and proxy caching, it seems realistic to adapt such algorithms from DSM to proxy caching, or vice versa. However, before improving one of the systems by studying the corresponding mechanism on the other, we must first study the viability the different natures of both systems causes.

Similarities are a double-edged sword. By selecting the cutting edge, we can find a good new solution; but selecting the wrong edge, we can uncover new problems.

ACKNOWLEDGMENTS

We thank the anonymous referees for their constructive comments, which helped improve this work's quality.

Spanish grant GV98-14-47 supported this work.

References

1. M. Wilkes, "Slave Memories and Dynamic Storage Allocation," *Trans. IEEE*, Vol. EC-14, 1965, p. 270.
2. V. Milutinovic and M. Valero, "Cache Memory and Related Problems: Enhancing and Exploiting the Locality," *IEEE Trans. Computers*, Vol. 48, No. 2, Feb. 1999, pp. 97-99.
3. V. Milutinovic, *Microprocessor and Multi-microprocessor Systems*, John Wiley & Sons, New York, 2000.
4. V. Milutinovic, "System Support for Electronic Business on the Internet," <http://galeb.etf.bg.ac.yu/~vm/books/2001/ebi.html> (current July 2000).
5. I. Tartalja and V. Milutinovic, *The Cache Coherence Problem in Shared-Memory Multiprocessors: Software Solutions*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996.
6. V. Milutinovic and P. Stenström, "Scanning the Issue," *Proc. IEEE*, Vol. 87, No. 3, Mar. 1999, pp. 399-404.
7. M. Abrams et al., "Caching Proxies: Limitations and Potentials," *Proc. Fourth Int'l World Wide Web Conf.*, Boston, Mass., 1995, pp. 119-133.
8. J. Protic and V. Milutinovic, *Distributed Shared Memory: Concepts and Systems*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1998.
9. P. Cao and C. Liu, "Maintaining Strong Cache Consistency in the World Wide Web," *IEEE Trans. Computers*, Vol. 47, No. 4, Apr. 1998, pp. 445-457.
10. J. Gwertzman and M. Seltzer, "WWW Cache Consistency," *Proc. 1996 Usenix Technical Conf.*, Usenix Assoc., San Diego, Calif., 1996, pp. 141-151.

11. G.T. Byrd and M.J. Flynn, "Producer-Consumer Communications in Distributed Shared Memory Multiprocessors," *Proc. IEEE*, Vol. 87, No. 3, Mar. 1999, pp. 456-466.
12. A. Dingle and T. Partl, "Web Cache Coherence," *Proc. Fifth Int'l World Wide Web Conf.*, 1996; www5conf.inria.fr/fich_html/papers/P2/Overview.html.

Juan-Carlos Cano is an assistant teacher in the Department of Computer Engineering at the Polytechnic University of Valencia (UPV) in Spain. His current research interests include novel mechanisms to reduce the volume of data for downloading and wireless systems. He earned an MS in computer science from the UPV, where he is currently a PhD candidate. Contact him at the Departamento de Informatica de Sistemas y Computadores, Escuela Universitaria de Informatica, Universidad Politecnica de Valencia, Cno. de Vera s/n, 46071 Valencia, Spain; jucano@disca.upv.es.

Ana Pont is an assistant professor in the Department of Computer Engineering at the UPV. Her research interests include multiprocessor architecture, memory hierarchy design, and performance evaluation. She received her MSc in computer science and her PhD in computer engineering from UPV. She is a member of the IEEE and ACM. Contact her at the Departamento de Informatica de Sistemas y Computadores, Escuela Universitaria de Informatica, Universidad Politecnica de Valencia, Cno. de Vera s/n, 46071 Valencia, Spain; apont@disca.upv.es.

Julio Sahuquillo is an assistant professor in the Department of Computer Engineering at the UPV. His current interests include multiprocessor systems, cache management, and instruction-level parallelism. He received his BS, MS, and PhD in computer science from UPV. Contact him at the Departamento de Informatica de Sistemas y Computadores, Escuela Universitaria de Informatica, Universidad Politecnica de Valencia, Cno. de Vera s/n, 46071 Valencia, Spain; jsahuqui@disca.upv.es.

José-Antonio Gil is an assistant professor at the UPV. His interests include distributed shared memory systems and proxy cache systems. He received his BS, MS, and PhD in computer science from the UPV. He is a member of the IEEE. Contact him at the Departamento de Informatica de Sistemas y Computadores, Escuela Universitaria de Informatica, Universidad Politecnica de Valencia, Cno. de Vera s/n, 46071 Valencia, Spain; jagil@disca.upv.es.