Parallel Pull-Based LRU: a New Request Distribution Algorithm for Large Scale SCI Clustered Web Caches

Emmanuel Cecchet, Cyril Perrin
INRIA Rhône-Alpes - SIRAC laboratory
{emmanuel.cecchet, cyril.perrin}@inrialpes.fr

Abstract

The SIRAC laboratory has developed SciFS, a Distributed Shared Memory (DSM) tightly integrated with the Linux operating system kernel, that exploits the high performance and remote addressing capabilities of SCI (Scalable Coherent Interface) networks. We experiment with using SciFS to build large scale clustered web caches. An efficient DSM does not only provide a fast and large data storage space, but it also allows the design of new algorithms to handle user requests.

We propose a clustered web cache prototype with a new request distribution algorithm, called PPBL (Parallel Pull-Based LRU), especially designed for use with a DSM system. Unlike other distribution algorithms the decision is distributed over all nodes thus providing better scalability. In this short paper, we only present PPBL principles but additional information can be found on our website at http://sci-serv.inrialpes.fr.

1. Introduction

Web caches have become the standard method to ensure an acceptable quality for Web access. For a cache implemented on a single dedicated computer, the throughput is quickly limited by the network interface. SMP computers also have to face this limitation. In contrast, cluster-based Web caches allow the throughput to be increased by simply adding nodes (and therefore network interfaces) to the cluster.

The distributed shared memory paradigm allows us to write distributed programs as in a centralized system. We no longer have to care about low level communications and data transfers. We use SciFS [2] to build a clustered web cache prototype that uses the DSM to provide large and fast data storage. With such an architecture, data placement is no longer a problem and traditional request distribution algorithms like simple round-robin or the more elaborated Locality-Aware Request Distribution (LARD) [1] can be enhanced by using new parallel strategies.

In the next section, we introduce the architecture of our prototype using SciFS. Then we describe PPBL, our new request distribution algorithm and finally we conclude.

2. Prototype architecture

Our prototype uses three different components: a router, a frontend, and several backends. Figure 1 shows the flow of a typical request. The router accepts a client connection (arrow numbered 1). The HTTP request is forwarded unmodified to the frontend node (2) that parses the request and looks for the requested object in the cache. The request is then forwarded to a backend (3) through the DSM (i.e. the SCI network), according to a given request distribution algorithm.

If it is a cache miss, the backend node opens a connection to the origin server (4). It sends a request and receives the HTTP reply. The reply header is parsed and selected elements are stored in the cache information structure. The received data are written in the DSM. At the same time, this data are sent back to the router (5) that forwards it to the client (6).

Figure 1. Flow of a typical request
If the request generates a cache hit, the backend node reads the data from the DSM and sends it back to the client through the router (5, 6).

We use an URL table in which we store the URL name and some other information related to the HTTP header. Each entry also contains a small three slot LRU cache, called serving LRU, providing the identifier of the last nodes that served this request. Thus, if a node is overloaded, the node itself can choose another node in this LRU to serve the request and preserve at best the data locality.

3. Parallel Pull-Based LRU

We propose a new approach to distribute incoming requests to backend nodes. Existing decision algorithms are executed on the frontend node that takes the decision and sends the request to the chosen backend node [1]. To reduce the bottleneck induced by this centralized solution, distribution algorithms have to stay simple to be quick, but this is not scalable.

It is possible to parallelize the decision algorithm on all nodes. We believe that such an algorithm can sustain higher request peaks which is essential for large scale web caches. We propose to put all the incoming requests in the DSM to be treated in parallel by the backend nodes. Instead of having one frontend pushing the requests to the backends, our approach allows the backends to pull the requests from the frontend. To prevent costfull coherency updates of the URL table, we distribute the table among the backend nodes. Each node manages its own part of the URL table using a local LRU. We call this request distribution algorithm Parallel Pull-Based LRU (PPBL).

Figure 2 describes the PPBL principle. A shared memory segment containing the incoming requests is fixed on the frontend node. Each backend node allocates a LRU linked list in its local memory and maps a part of the URL table in the DSM. Each mapped part is different and aligned on page boundaries to prevent any false sharing, migration or any mechanism used by the DSM to keep the table coherent between all nodes. Using this scheme, every access is local to each backend node.

Each backend node picks up a not yet found URL request in the incoming queue, and searches its URL list to check if it owns the URL or not. The lookup is made in the LRU order. If the matching entry is found, the backend signals that it has found the URL by updating the incoming request queue entry with its node identifier and the index of the URL in its table. If the backend accepts to serve the request, it starts sending the reply directly to the client.

At this point, the frontend can then just discard the entry from the incoming queue. It is possible to use a signaling mechanism to interrupt the lookup on other nodes as soon as one node has found the requested URL.

If the entry is not found, we have to add it to the URL list and get it from the web. The node that will do the job is chosen using a simple round robin unless it notices that it is overloaded. In this case, the node is temporarily evicted from the round robin. If the adding URL distribution among the backend is equitable, then the distributed LRU has the same behavior as a centralized one.

4. Conclusion

Traditional clustered web caches request distribution algorithms do not scale well because the decision is centralized on one node, i.e., the frontend. Therefore, we propose a new algorithm, called Parallel Pull-Based LRU (PPBL), especially designed for use with an efficient DSM. Unlike usual algorithms, the decision is not centralized on the frontend, but instead the backends take the incoming requests directly on the frontend. In this way the request treatment can be easily parallelized and thereby reducing considerably the load on the frontend.

Very early results confirm that PPBL is an efficient and scalable distribution algorithm for clustered web caches built over a DSM system. However scalability issues can quickly arise if the implementation uses too heavy synchronization between the frontend and the backend nodes.

PPBL is distributed as part of the SciFS Test Suite and is freely available for download and evaluation from http://sci-serv.inrialpes.fr.

5. References
