1. Introduction

Distributed shared memory (DSM) (Kai Li 1986. Kai Li and P. Hudak 1989) is a software abstraction allowing a set of workstations connected by a LAN to share a single virtual address space. One of the key problems in building any software DSM is minimizing the amount of data communication required to keep the distributed memories consistent.

Munin is a DSM based on software objects (variables) but uses the workstation virtual memory (V kernel) to detect access to the shared objects. Munin includes several techniques to reduce consistency-related communication, namely

a) software release consistency, a technique that requires the memory to be consistent at specific synchronization points,

b) multiple consistency protocols, to allow the user to select the best consistency protocols for each data item,

c) a write-shared protocol, to reduce the problem of false sharing,

d) an update-with-timeout mechanism to provide direct updates without invalidation of active data items.

Most of these features require programmers to annotate their shared variables. Variables that are not annotated are treated as conventional shared variables (see below). Incorrect annotations may result in inefficient performance or in runtime errors that will be detected by the runtime system but not in incorrect results.

Munin includes:

a) a preprocessor that reads the user annotations and converts them into compiler directives controlling the placement of the shared data in virtual memory

b) a run-time system that maintains an object directory and a delayed update queue.

2. Software release consistency

Munin is based on a software implementation of release consistency that distinguishes three kinds of variables:

a) ordinary variables that can only be accessed by the process that created them;

b) shared data variables that are visible to all Munin processes and will appear sequentially consistent as long as they are always accessed from within critical regions;

c) synchronization variables, locks, barriers, or condition variables; that must be accessed through special library procedures such as lock() and unlock() for locks.

When a process modifies shared data inside a critical region, all update messages are buffered and delayed until the process leaves the critical region (eager release).

3. Multiple consistency protocols

Munin offers four consistency protocols aimed at supporting different types of access patterns:

a) conventional shared variables: they are replicated on demand and kept consistent using an invalidation-based protocol requiring a writer to be the sole owner of the data it wants to modify;

b) read-only variables: they cannot be modified once they have been initialized and are replicated on demand; any attempt to modify them will result in a runtime error;

c) migratory variables: they are migrated among the processes accessing them; every process accessing a migratory variable will always get full read and write access even if it only requested read access;

d) write-shared variables: they can be updated concurrently without intervening synchronization because the programmer knows that different portions of the data are accessed.

4. Write-shared protocol

Since Munin relies on the virtual memory to control all accesses to the DSM, the implementation of its write-shared protocol is based on a copy-on-write mechanism. Whenever a process is granted access to write-shared data, the page containing these data is marked copy-on-write. Hence the first attempt to modify the contents of the page will result in an interrupt that will cause the creation of a copy of the page (the twin).

At release time, the DSM will perform a word by word comparison of the page and its twin, store the diff in the space used by the twin pages and notify all processes having a copy of the shared data of the update.

A runtime switch can be set to check for conflicting updates to write-shared data.

5. Update Time-Out Mechanism:

To reduce the cost of sending update messages, Munin does not send updates to processes holding stale replicas. Anytime a process receives an update for a page for which it does not have a twin, the page is marked supervisor-only, and the time of receipt of the update is recorded.

The first local access to the page will cause a trap that will remove the restriction. When a process receives an update for a page that is still marked supervisor only, it checks the timestamp of the last update to the page. If more than 50 ms have elapsed it notifies the originator of the update not to send more updates and invalidates the page.

6. Conclusion

The strongest point of Munin is its excellent performance, typically within 5 to 33% of that of message passing versions of the same programs. (Its major limitation is its dependence of some features of the V kernel, such as allowing user-level programs to catch page access violation interrupts.)

7. Further Developments

The same team has come with a successor to Munin named TreadMarks. While Munin uses an eager release protocol and resynchronizes the DSM after every release, TreadMarks uses a more complex lazy release protocol that delays updates. Unlike Munin, TreadMarks is UNIX based.