A scheme to combine the benefits of strong and weak memory consistency systems is suggested, by embedding Linda-like tuple locks in shared objects to act as control indicators.

Keywords: strong and weak memory consistency, Linda tuple locks,

CR Categories: B.3.2 – Cache memories, Shared memory, D.4.1 – Concurrency, Deadlocks.

1. INTRODUCTION
The issue of memory consistency arises whenever multiple copies of shared information are handled by separate processes. Specifically, the question amounts to: when will modifications made by one process become visible to other processes? If the answer is “immediately”, then we have what is termed “sequential memory consistency”. The cache coherence protocol adopted by shared memory multiprocessors is an example of such a memory consistency mechanism, with write operations in one cache causing invalidation of copies of the same memory block in other caches, hence causing future reads on such locations to obtain fresh copies that contain the latest modifications.

Sequential memory consistency is considered to be a “strong” consistency, and it is quite costly to provide, particularly on distributed systems where the runtime system has to enforce it between multiple copies held in the memories of different processors; weaker memory consistency models, which retreat from a guarantee of immediate visibility of writes on one copy to the other copy holders, aim to improve efficiency without hampering the effectiveness of parallel programming systems. By delaying the forwarding of write results to readers until certain “strategic moments”, multiple modifications are buffered and consolidated, so that only the final result is forwarded once, rather than multiple modifications being forwarded individually as they occur.

For example, a process would lock a shared page, perform a number of writes on it, and unlock, without other processes becoming aware of modifications having been made on the shared information; but when another process requests a lock on the same page, it would be notified of the changes and would then acquire the information through a standard system mechanism. The cost of memory sharing may be further reduced by exploiting the virtual memory mechanism: when a lock on a group of pages is acquired, instead of forwarding new page contents, the system merely asks the process that acquires the lock to invalidate its own shared pages, so that new contents are forwarded only when a page is actually accessed, while avoiding transferring pages that are not actually used.