Multi-tier Data Synchronization Based on An Optimized Concurrent Linked-list

Bing Yang, Kenneth B. Kent, Eric Aubanel
University of New Brunswick, Faculty of Computer Science
Tobi Agila, Angela Lin
IBM Canada
{bing.yang, ken, aubanel}@unb.ca {atobia, angela_lin}@ca.ibm.com

Introduction

The development of multicore processors in recent years raises new challenges in the development of efficient approaches to allow concurrent threads to access shared resources safely. *PackedObject* is a experimental enhancement introduced in IBM's J9 Virtual Machine. It organizes data in a multi-tier manner (like a C struct) in which the object data is nested in its enclosing *PackedObject* instead of being pointed to by an object reference. This multi-tier data structure brings the need and new challenges for synchronization that allows multiple threads to lock on the multi-tier data from different tiers and maintain consistency.

Multi-tier PackedObjects

- Packed object data model organizes data in a multi-tier manner.
- Each *PackedObject* has an object header containing the start address (target+offset) of the packed data.

```
@Packed class OneTierClass extends PackedObject
{
   int value;
}
@Packed class TwoTierClass extends PackedObject
{
   OneTierClass D;
   OneTierClass E;
}
@Packed class ThreeTierClass extends PackedObject
{
   OneTierClass B;
   TwoTierClass C;
}
ThreeTierClass A = PackedObject.newPackedObject(ThreeTierClass.class);
```

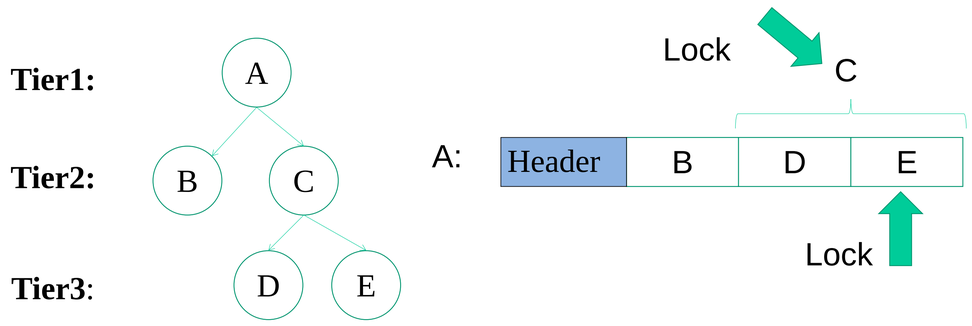

Figure 1: Multi-tier Data structure of PackedObjects

Challenges

- Multiple threads can arbitrarily access any level of the multi-tier *PackedObject* concurrently.
- Multiple threads can arbitrarily update the data overlapped in different tiers.
- Different threads are unaware of the containing relationship of the data in different tiers

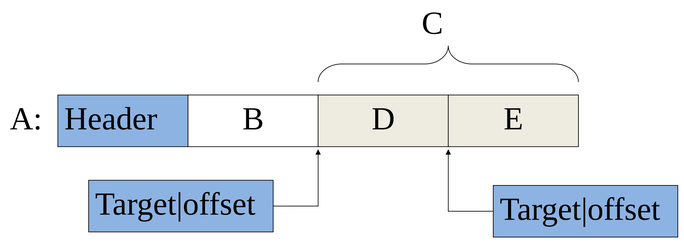

Figure 2: Memory layout of derived and non-derived PackedObject

Multi-tier Data Synchronization Approach

- Use a linked list to keep track of the monitor information, PackedMonitorInfo (PMI), for *PackedObjects* that share the same packed data region.
- Multiple threads locking on the same packed data region traverse the same linked list to find the associated monitor.

Figure 3: A monitorList associated with the PackedObject A

- Build a concurrent linked-list based on the lazy-list algorithm
- Optimize the lazy-list algorithm, mainly in two aspects:
  - Postpone the physical deletion of nodes by using a predetermined threshold to reduce unnecessary repetition of insertion/deletion.
  - Reverse the locking order to reduce locking and traversal overhead when the validation of pointers fail.

Future Work

- Explore other data structures (like trees) that could better represent the containing relationship of different tiers of data.
- Build a multi-granularity locking scheme based on the new data structure.