Interval Based Framework for Locking in Hierarchies

Saurabh Kalikar and Rupesh Nasre

{saurabhk, rupesh}@cse.iitm.ac.in
PACE Lab, CSE, IIT Madras

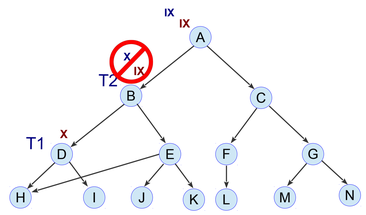
PACE
Programming Languages, Architecture and Compilers Education Laboratory

Introduction:

- We present efficient locking mechanisms for hierarchical data structures.
- A hierarchy is characterized by a containment relationship, in which the child nodes are contained within their parent nodes.
- Fine-grained locking Vs Hierarchical locking.
  - Fine-grained locking: Lock at node B only protects nodes B.
  - Hierarchical locking: Lock at node B protects whole sub-hierarchy rooted at B.

Challenges:

- Consider a scenario where thread *T1* has acquired a lock on a node *D*.
- Now, another thread *T2* wants to acquire a lock on a node *B*.
- As node *D* is contained in the structure rooted at node *B*, thread *T2* can also access node *D* which can lead to the data-race.
- Node *D* and *B* can not get locked simultaneously because of such hierarchical dependency.
- Intention locks are widely used for hierarchical locking. X represents exclusive lock and IX is corresponding intention-exclusive lock.

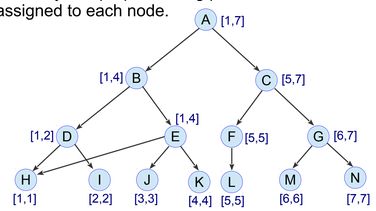


Hierarchical Locking using Intervals:

- We address the practical issues in Intention Locks (IL) and present a novel approach of *hierarchical locking using intervals*.
- We assign an interval value to every node in the hierarchy in a preprocessing phase.
- An interval is a pair **(low, high)** of integer values assigned to each node.

Properties of Logical Intervals:

- If the Intervals of two nodes subsume, then they have ancestor-descendant relationship.
- If the Intervals of two nodes partially overlap, then they have at least one common descendant node.
- If the Intervals of two nodes are non-overlapping, then the hierarchies rooted at these nodes are disjoint.



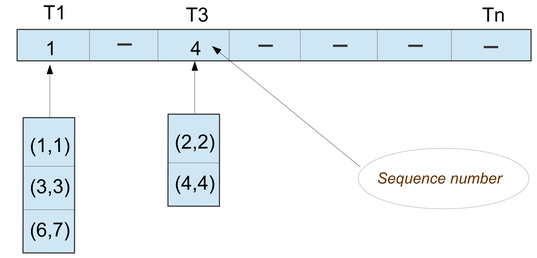
Interval of B subsumes interval of D representing parent-child relation

Lock Manager:

Hierarchical Locking Using Concurrent Pool:

- Maintains locks in the form of numbered intervals.
- Unlike traditional hashed index of resources and waiting queues, we index our pool of locks according to thread-ids.
- Every thread has a specific location where it inserts its lock entries and each thread gets a unique sequence number before accessing lock manager by incrementing a global counter.
- Incrementing the counter and inserting the set of intervals to be locked happen atomically.
- The lock intervals for each thread are maintained in sorted order to avoid deadlocks.
- After insertion, each thread checks independently whether there is any conflicting entry in the lock-pool having a smaller sequence number.
- If the thread does not find any such overlap, the lock on all the inserted intervals is granted.

Concurrent Lock Pool



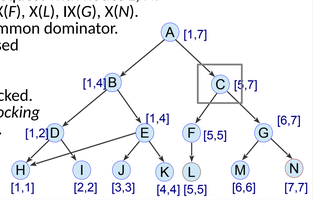
DomLock

Protocol:

- A node is allowed to get locked in *exclusive* mode, if the interval of the node does not overlap with any interval already locked in *shared or exclusive mode.*
- A node is allowed to get locked in *shared mode*, if the interval of the node does not overlap any interval already locked in *exclusive mode.*
- If locking request consists of multiple nodes, lock the *immediate common dominator* node of all requested nodes.

Example: Consider a locking request with nodes *L*, *N*.

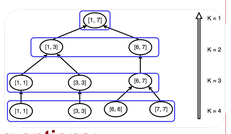
- Intention Locks: IX(*A*), IX(*C*), IX(*F*), X(*L*), IX(*G*), X(*N*).
- DomLock: Find immediate common dominator.
  - Lock C[5, 7] in interval based hierarchical locking.
  - Unit locking cost.
  - Extra node M is getting locked.
  - It is a trade-off between *locking cost* and *concurrency cost.*



NumLock:

- NumLock addresses the issues of DomLock by balancing the locking cost and concurrency cost.
- The locking request is represented as a set of intervals.
- NumLock generates *few* pareto-optimal locking option according to locking cost and concurrency cost.
- Consider locking request: H, J, M, N.

	Locking options	Locking cost	Concurrency cost
1	{H, J, M, N}	4	0
2	{G, H, J}	3	0
3	{E, M, N}	3	1
4	{B, C}	2	3
5	{E, G}	2	1
6	{A}	1	3



Cost model for comparing locking options:

- Locking cost: Regression function for the number of intervals locked.
- Size of critical section: Average of critical section sizes in past history.
- Contention index: Probability of the lock conflicts because of imprecise locking by a locking option.
- Number of parallel threads.

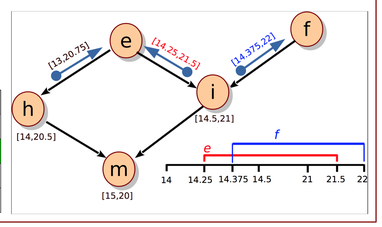
Hi-Fi:

- Hierarchical locking protocols do not support a fine-grained lock on internal nodes.
- Simple bottom-up interval numbering is not capable of identifying an internal node uniquely. For example, Node L and F.
- We propose a new locking protocol which supports both hierarchical and fine-grained locking semantics.
- We present a novel interval numbering which assigns unique interval to every node in the hierarchy.

Compatibility Matrix

Lock Requested by thread T2

Lock Held by thread T1	X, H	X, F	Y, H	Y,F
X, H	No	No	No	No
X, F	No	No	Yes	Yes
Y, H	No	Yes	No	No
Y,F	No	Yes	No	No

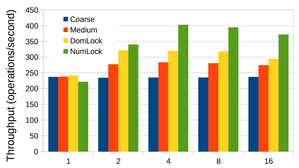


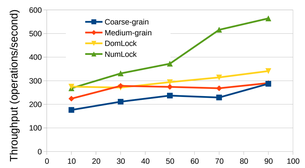
Ongoing and Future work:

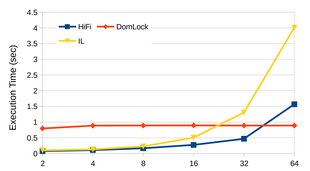
- The development of a hierarchical locking benchmark which provides a common platform for comparison between different hierarchical and fined-grained locking techniques.
- The benchmark supports operations with shared and exclusive accesses to the hierarchy with different skewness in access patterns.
- It allows to chose a certain locking protocol for the execution and provides command-line interface to configure various parameters such as sizes of critical section, the number of nodes in a lock request etc.
- Future work includes the design of a lock manager using concurrent interval trees and evaluate it against the lock managers in the real-world database systems.

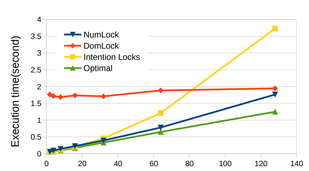
Experimental Evaluation:

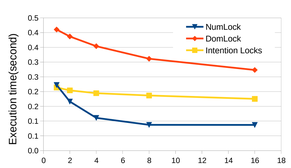
- Carried out on an Intel Xeon E5-2650 v2 machine.
  - 32 cores at 2.6 GHz, 128 GB RAM, CentOS 6.5.
- STMBench7: A benchmark to test the effectiveness of locking techniques and the STM implementations.
- STMBench7 has two existing locking techniques, Coarse-grained locking and Medium-grained locking.
- Stress Testing:
  - 1 million node binary tree and directed graph data structures, real-world XML hierarchy.
  - As we increase the number of lock requests per thread, the locking cost of Intention Locks increase linearly, However, DomLock shows constant locking cost.
- DomLock and NumLock both show higher throughput than coarse-grained and middle-grained locking in STMBench7.
- NumLock: On an average 25% throughput improvement over DomLock.
- NumLock scales well according to the percentage of read-only operations, skewness in the access patterns of locking requests, and number of locked nodes.
- Both NumLock and DomLock are incapable of handling fine-grained requests and treat every request as a hierarchical locking request.
- Hi-Fi shows gradual improvement in execution as we increase the percentage of fine-grained operations.

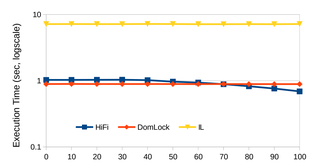












References:

- Saurabh Kalikar and Rupesh Nasre. 2016. **DomLock**: A new multi-granularity locking technique for hierarchies. In PPoPP'16 and a journal version in TOPC'17.
- Saurabh Kalikar and Rupesh Nasre. 2018. **Numlock**: Towards Optimal Multi-Granularity Locking in Hierarchies. In ICPP'18.
- Ganesh K, Saurabh Kalikar and Rupesh Nasre. 2018. **Hi-Fi:** Multi-Granularity Locking in Hierarchies with Synergistic Hierarchical and Fine-Grained Locks. To be published in EuroPar'18.
- Liu and Zang, 2016. Unleashing concurrency for irregular data structures. In ICSE'14.
- Gray, J.N., Lorie, R.A., Putzolu, G.R.1975: Granularity of Locks in a Shared Data Base. In VLDB'75

Ph.D. Forum: International Conference on Parallel Processing (ICPP 2018), University of Oregon, Eugene, Oregon, USA13-16 August, 2018.