CALock

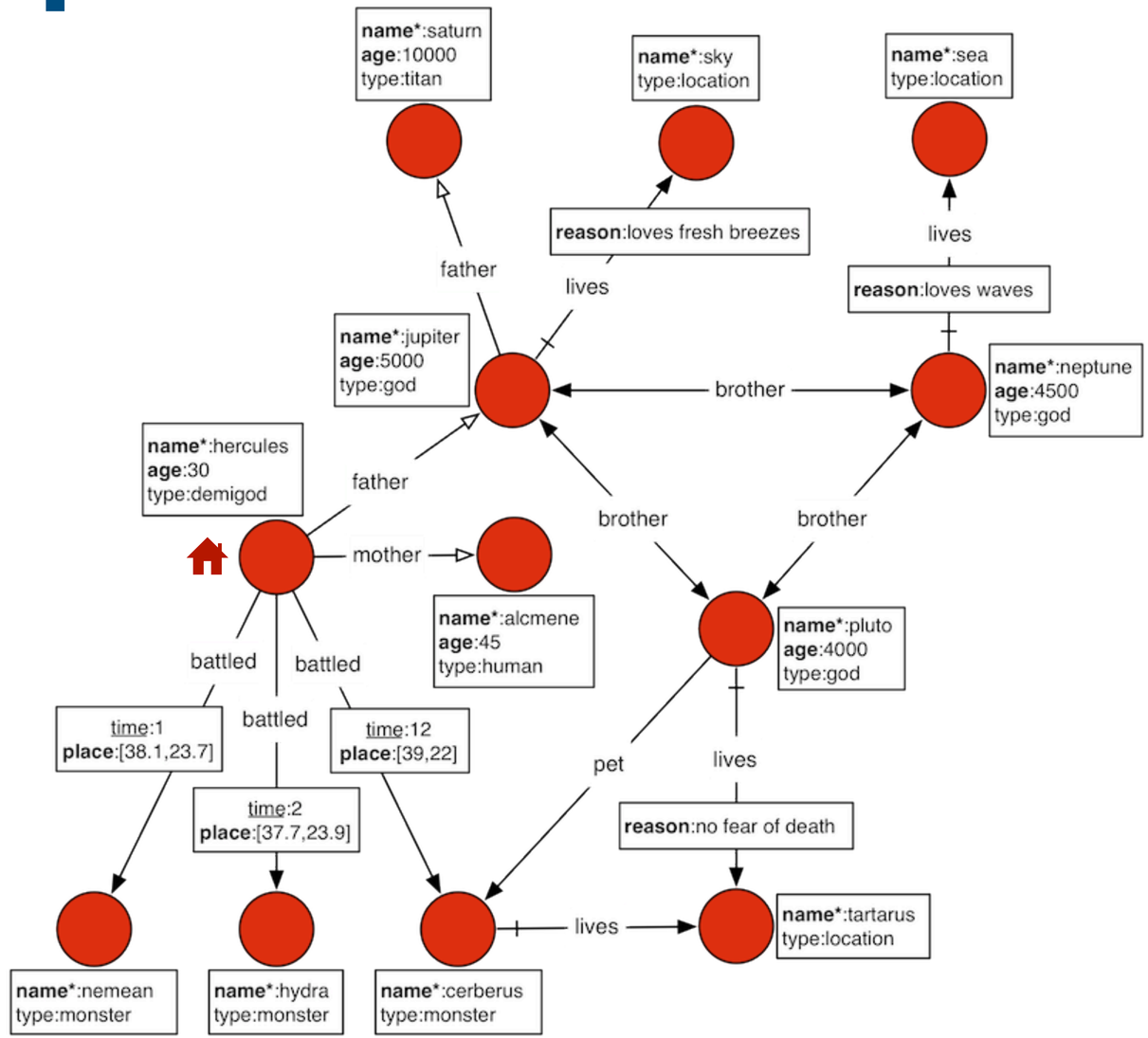
Verrouillage multi-granularité dans les graphes orientés

Ayush Pandey, Swan Dubois, Marc Shapiro, Julien Sopena

ComPAS 4-7 Juillet 2023

Connected Data - Graphs

- Vertices contain data objects
- Edges represent the relationships between data objects
- Traversals based on these relationships help answer queries

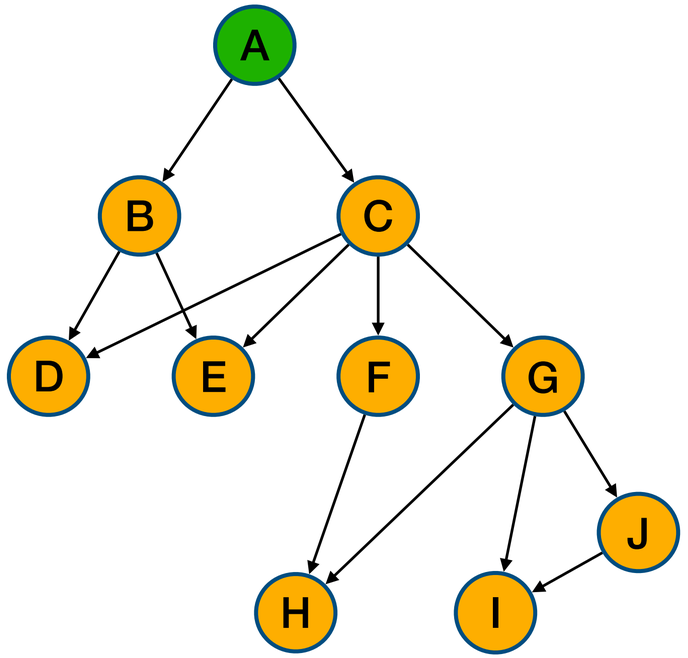


Thread synchronisation techniques

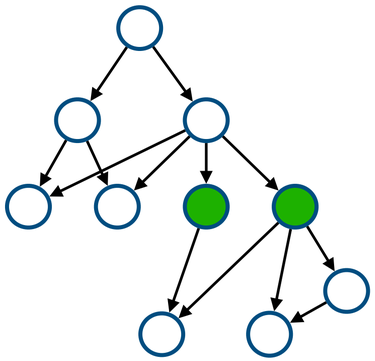
- Special underlying data-structures:
  - Lock-free trees, graphs
  - Conflict free datatypes i.e. CRDTs
- Special operation implementations:
  - Compare and Swap
  - Memory barriers and Operation reordering
- Primitives
  - Semaphores
  - Mutexes
  - Read/write locks

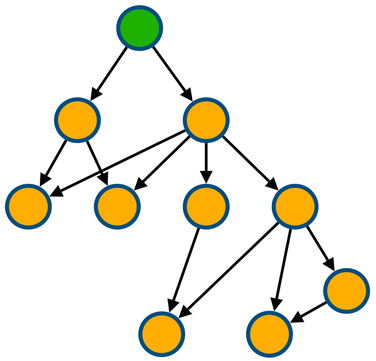
Locking in Oriented Graphs - Terminology

- Lock Target - Vertex which is locked
- Grain - Set of vertices guarded by this lock
- Granularity - Size of this grain



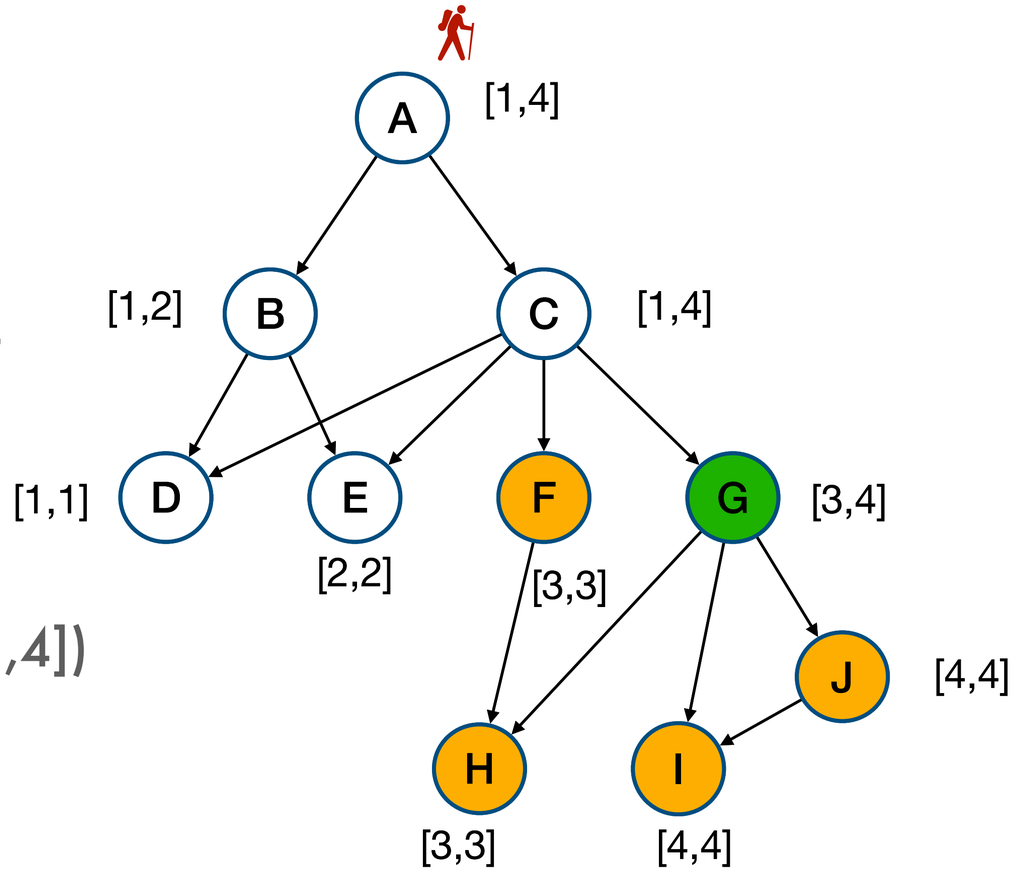
Lock granularities





MGL with DomLock

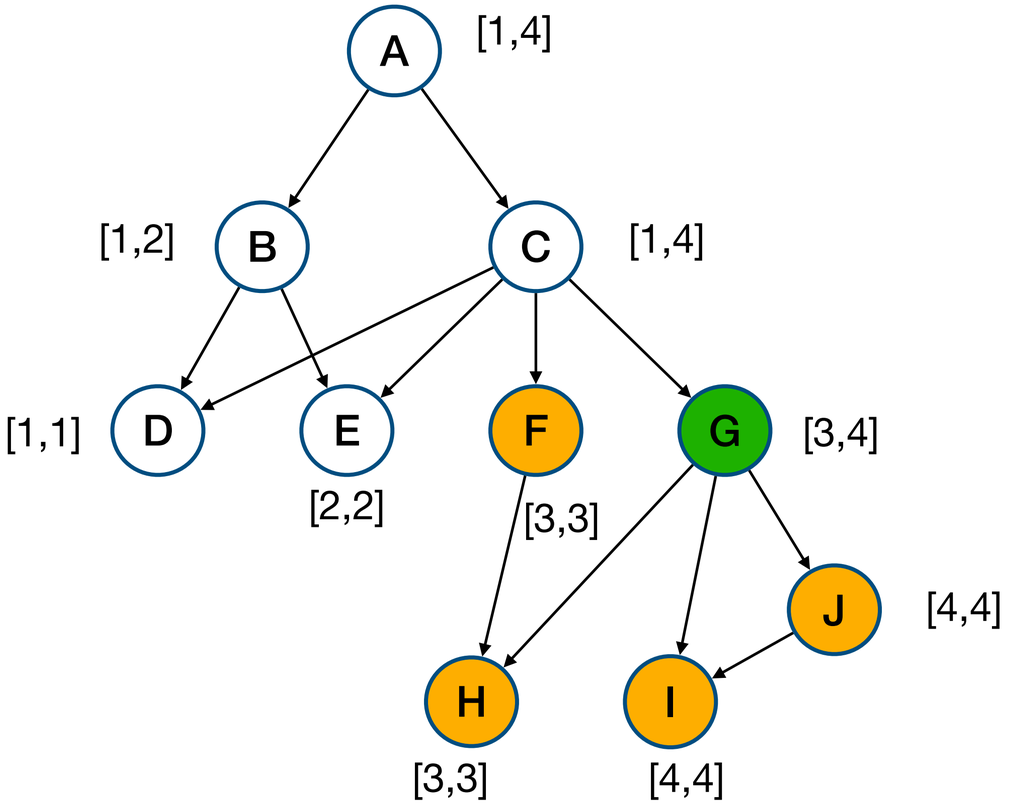
- Post-order traverse the graph
- Assign integer intervals to vertices in traversal order
- Use intervals to identify the lock grain and lock target



WriteLock(G,H) = WriteLock([3,3], [3,4]) = WriteLock([3,4])

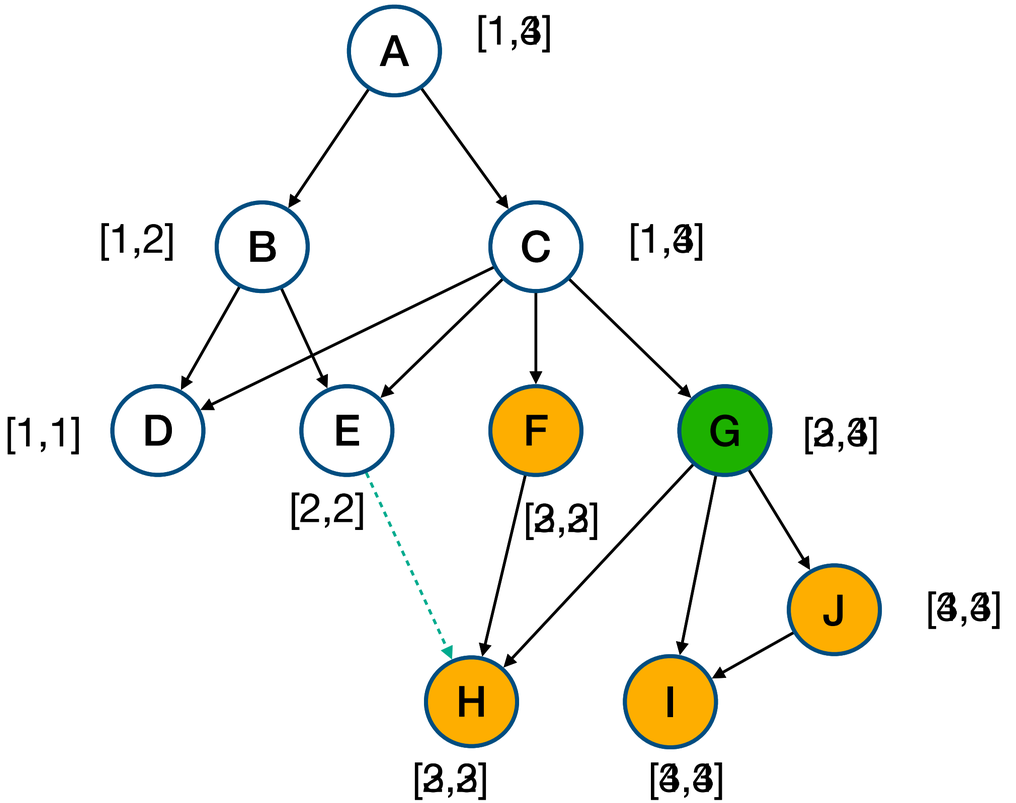
MGL with DomLock

- Lock target identification requires traversals
- False subsumptions might happen
- Intervals are not elastic



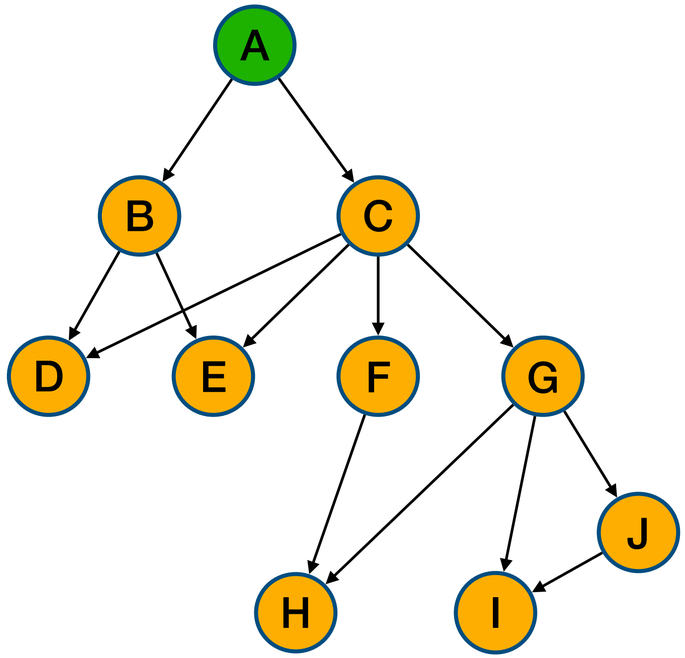
Problems with MGL state of the art

- Placing intention locks requires traversals
- Lock target identification requires traversals
- False subsumptions might happen
- Maintaining intervals is expensive



Constraints

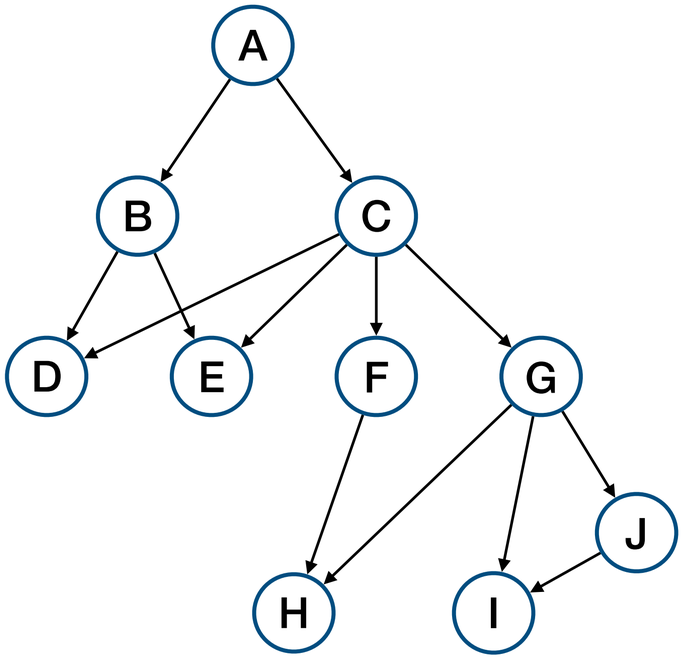
- Lock Target - Each thread holds one lock at any given time
- Graph has a single root



MGL using Common Ancestors - CALock

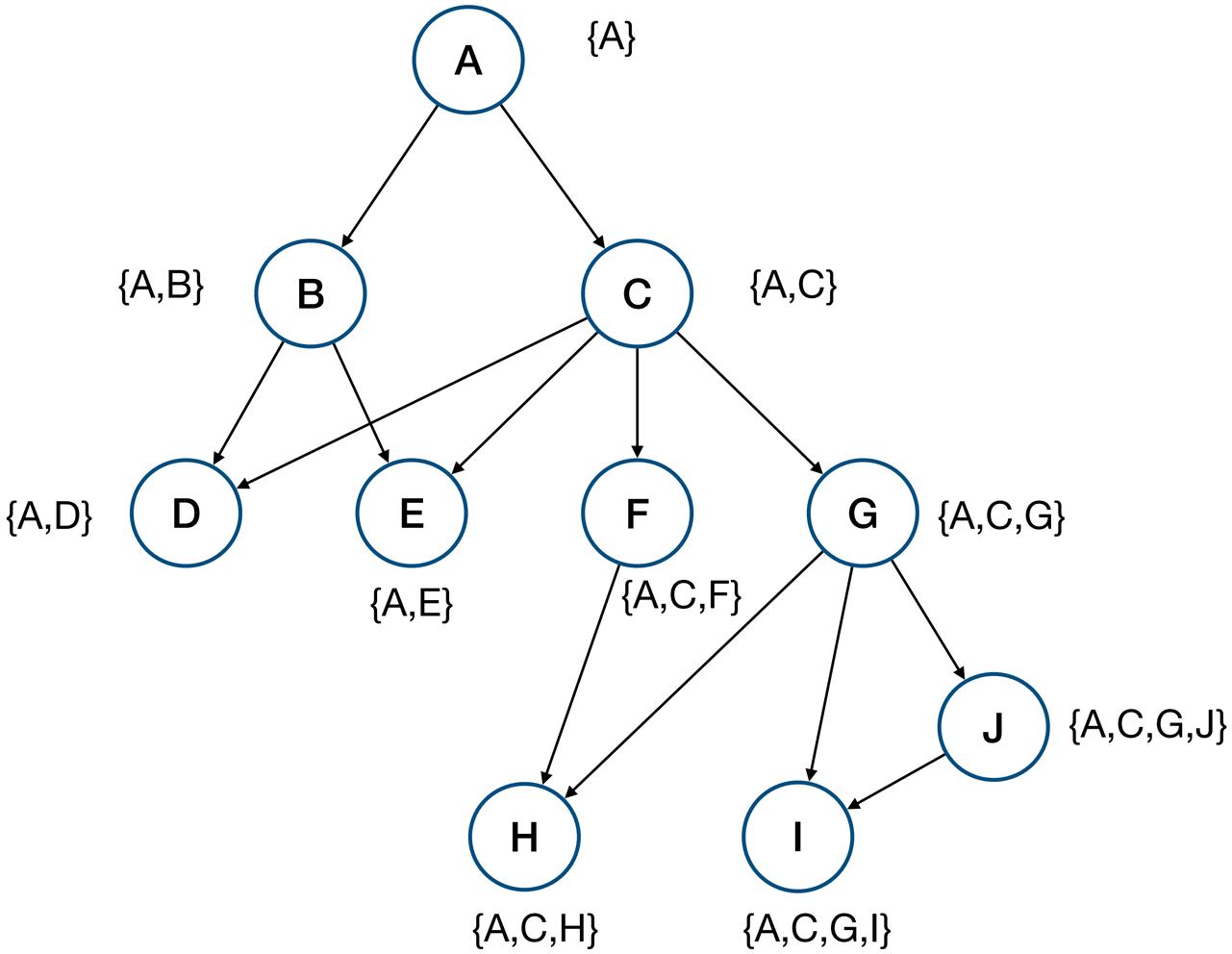
- Instead of using intervals to label vertices, CALock uses sets.
- The label set for any vertex is given by the relation:

$$L_v = \{v\} \cup \{ \bigcap_{u \in parents(v)} L_u \}$$



Working example of CALock Labelling

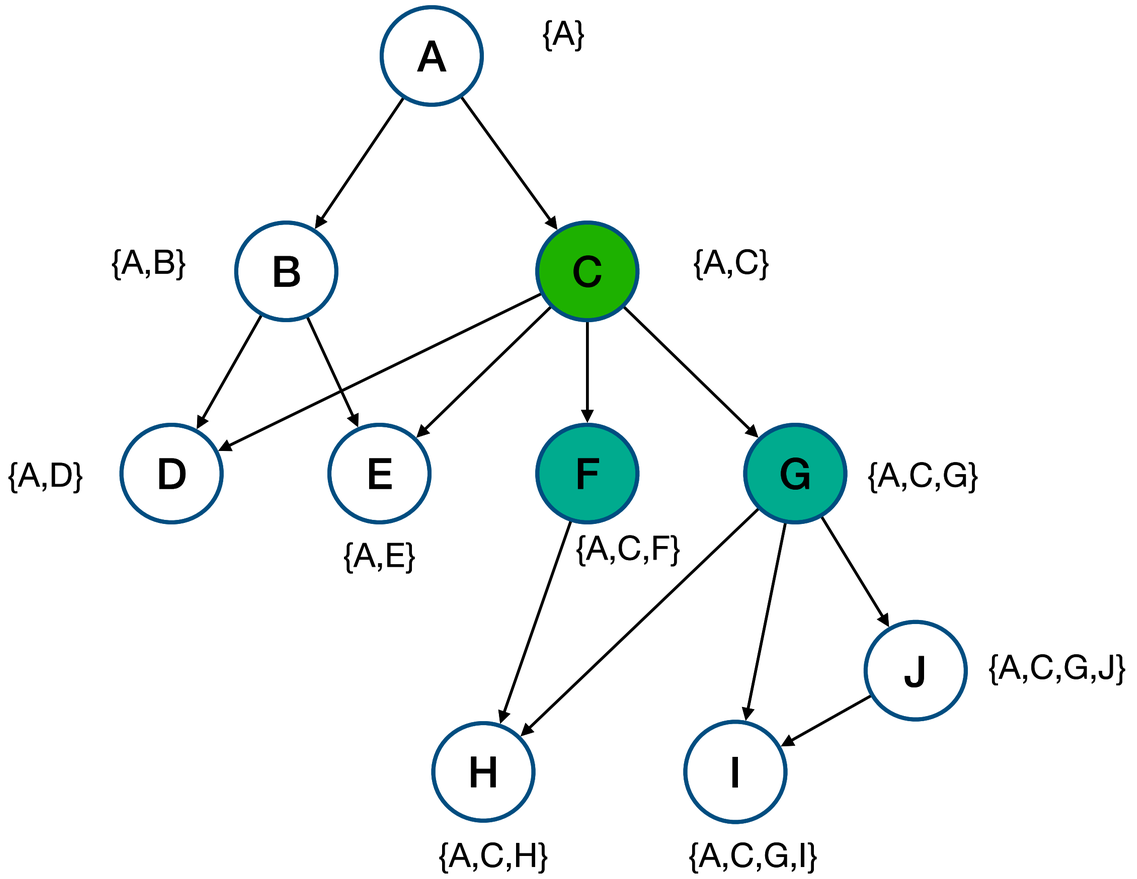
$$L_v = \{v\} \cup \{ \bigcap_{u \in parents(v)} L_u \}$$



Locking with CALock

- To lock $u, v,$ the Lock target is the deepest vertex in $L_u \cap L_v$

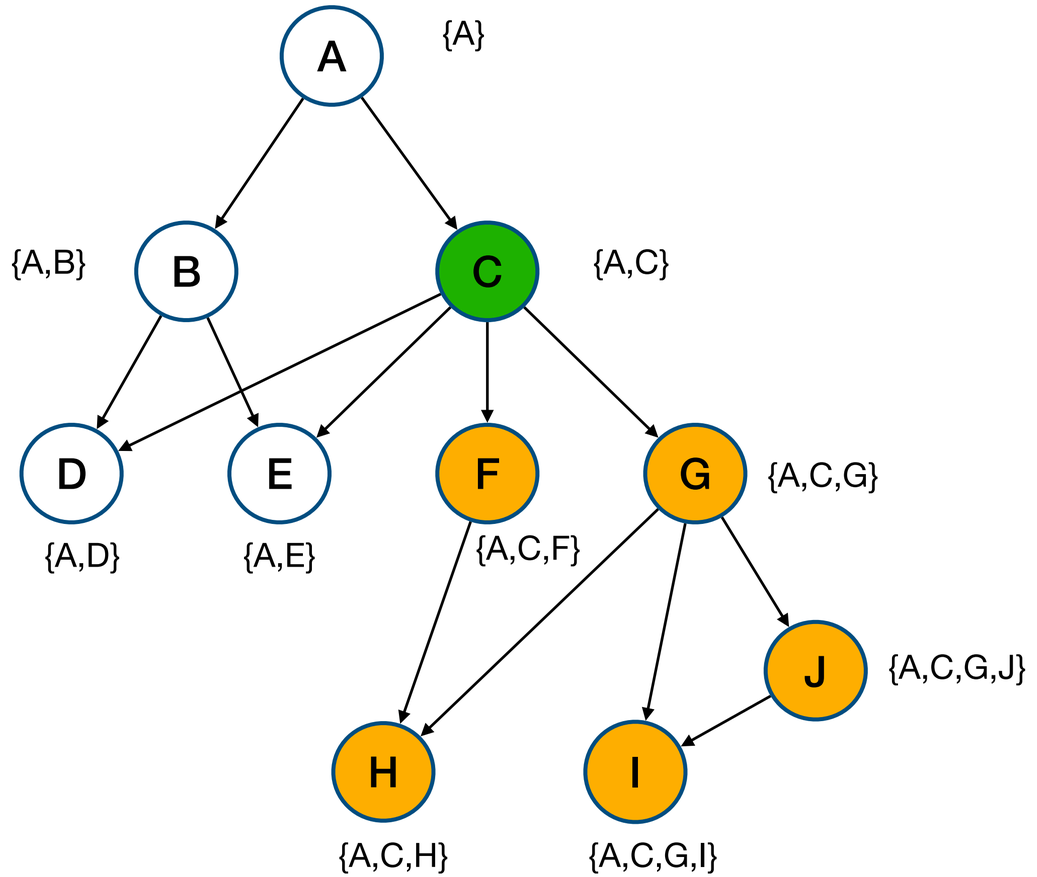
- Lock (F, G)
  - $L_F \cap L_G$ = {A,C}
  - Deepest node is C
- Place lock on C, which is the LSCA of F and G



Lock Grains with CALock

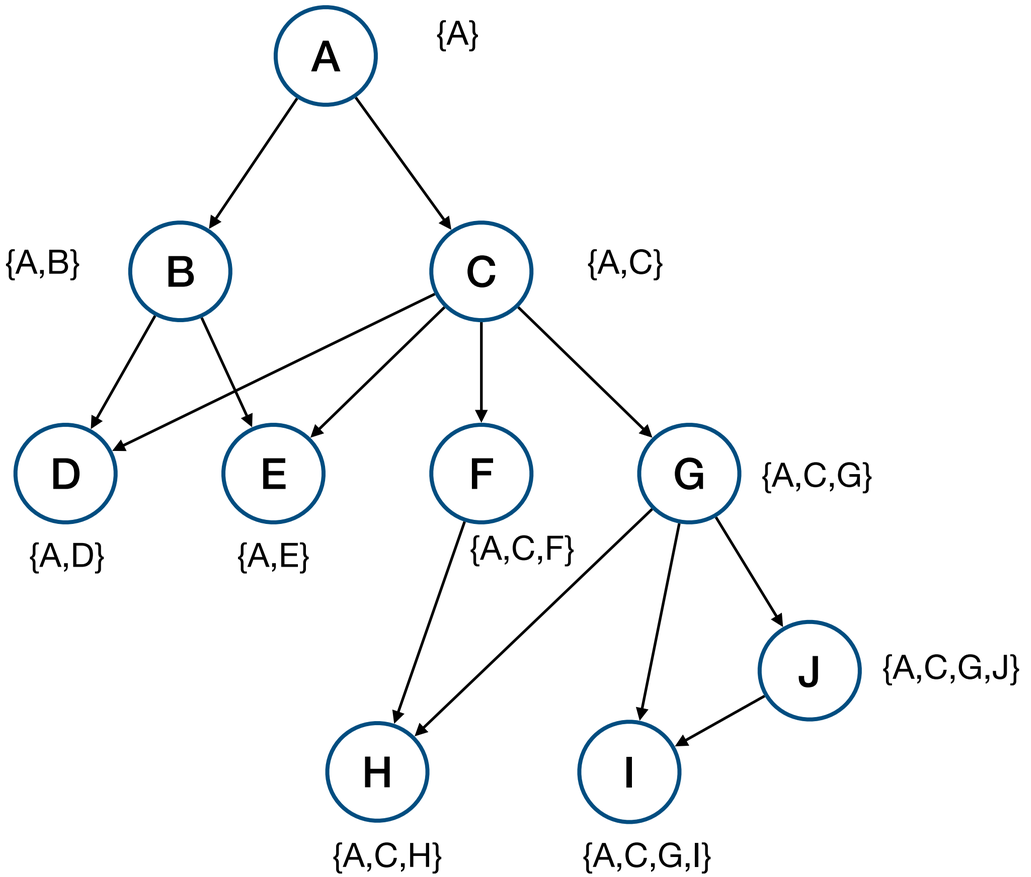
- The lock grain of a lock target u contains any vertex with u in its label.

- Grain of C = {C, F, G, H, I, J}



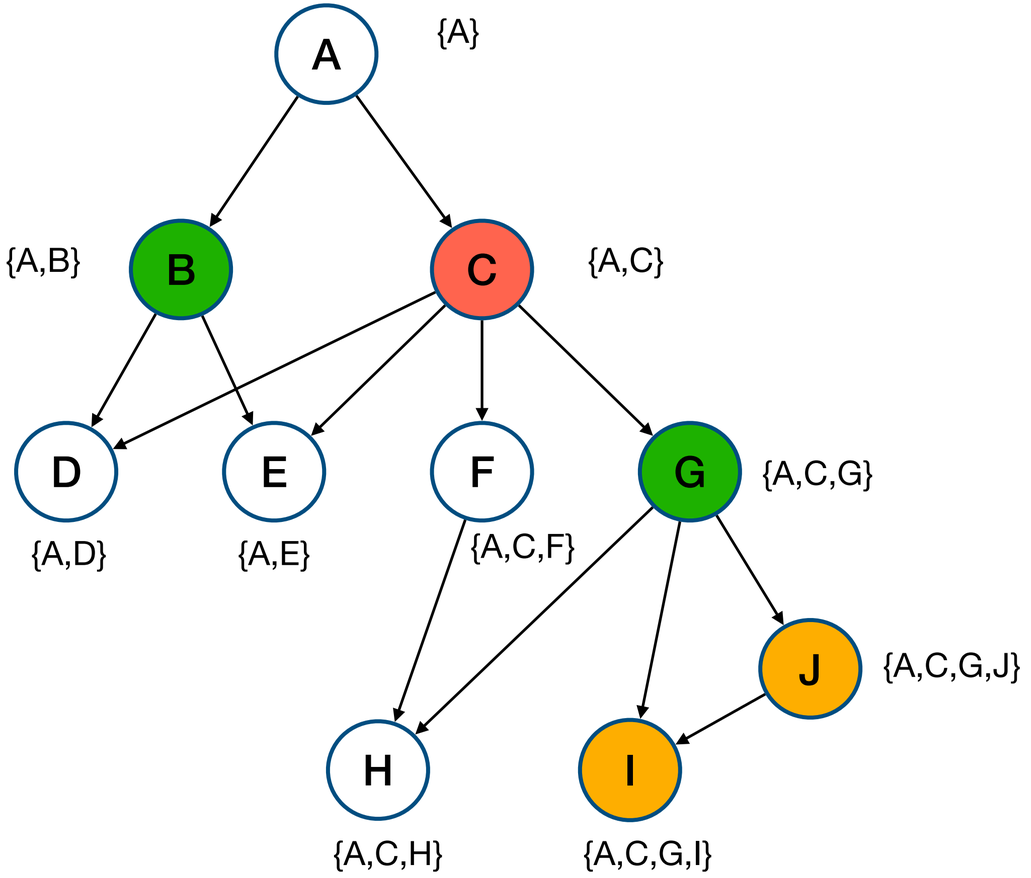
Concurrent Lock Pool

- Lock requests issued by threads are added to a pool
- The pool is a list of size N = number of threads
- A lock request in the pool contains:
  - A sequence number
  - Lock type
  - Lock target
  - Lock target label



Concurrent Lock Pool - Conflict detection

- T1 holds a lock on G, blocking I and J
- T5 holds a lock on B, blocking D and E
- T2 has requested a lock on C and checks for conflicts
- Threads conflict if these conditions hold simultaneously
  - Read/Write conflict
  - Lock grain overlap
  - Higher sequence number

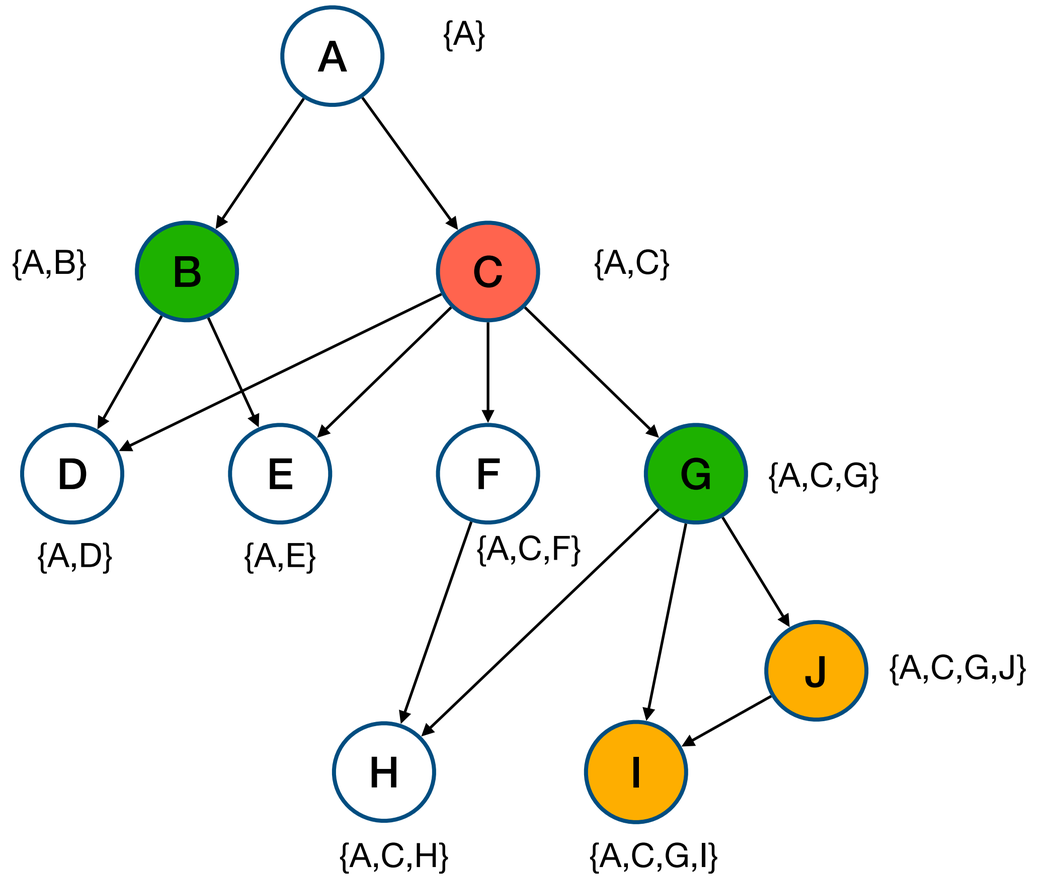


G, r, 2, {A,C,G}	C, w, 3, {A,C}	NULL	NULL	B, r, 1, {A,B}
T1	T2			T5

Concurrent Lock Pool - Conflict detection

- Read/Write conflict:
  - T2 has w and T1 has r => true
- Lock grain overlap:
  - T2 is locking C which contains G that is already locked
- Higher sequence number
  - T2 requested the lock after T1

T2 is blocked and waits for T1 to release the lock



G, r, 2, {A,C,G}	C, w, 3, {A,C}	NULL	NULL	B, r, 1, {A,B}
T1	T2			T5

Relabelling with CALock

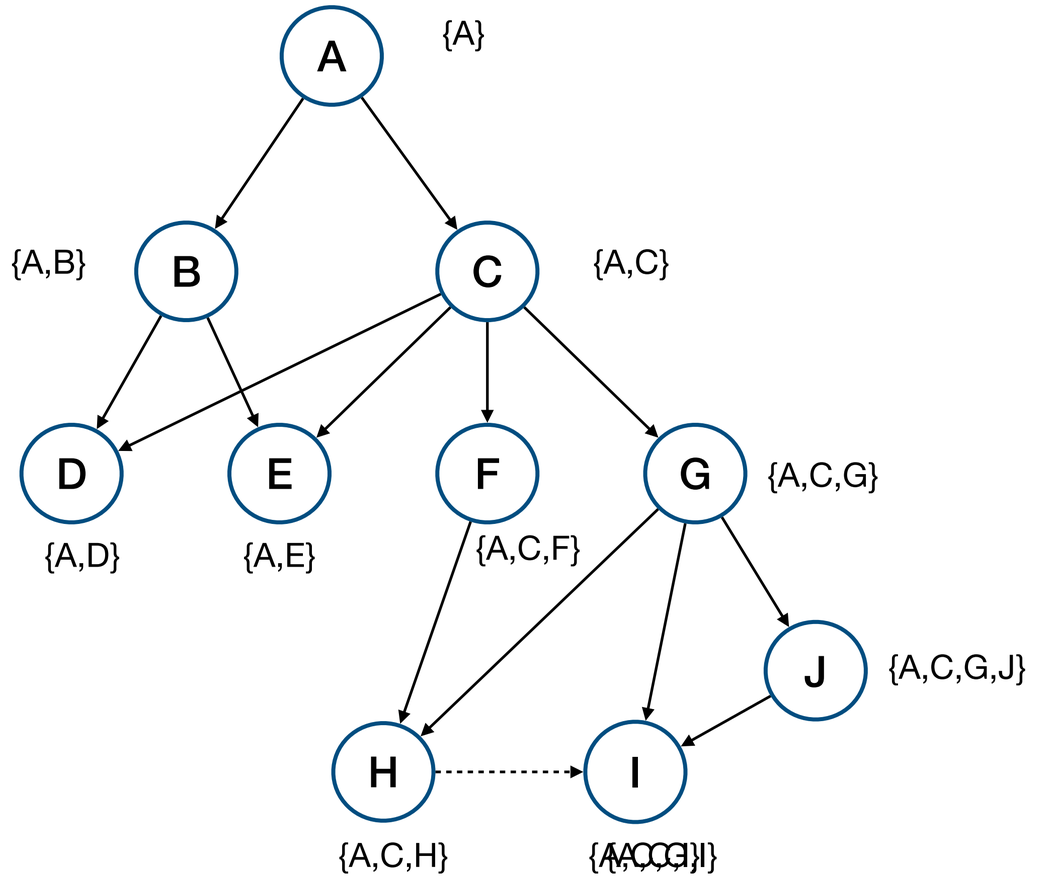
- Invoke the labelling relation recursively on the the affected vertex.

$$L_v = \{v\} \cup \{ \bigcap_{u \in parents(v)} L_u \}$$

- To add an edge between H and I, we take a lock on C and then start the relabelling at I.

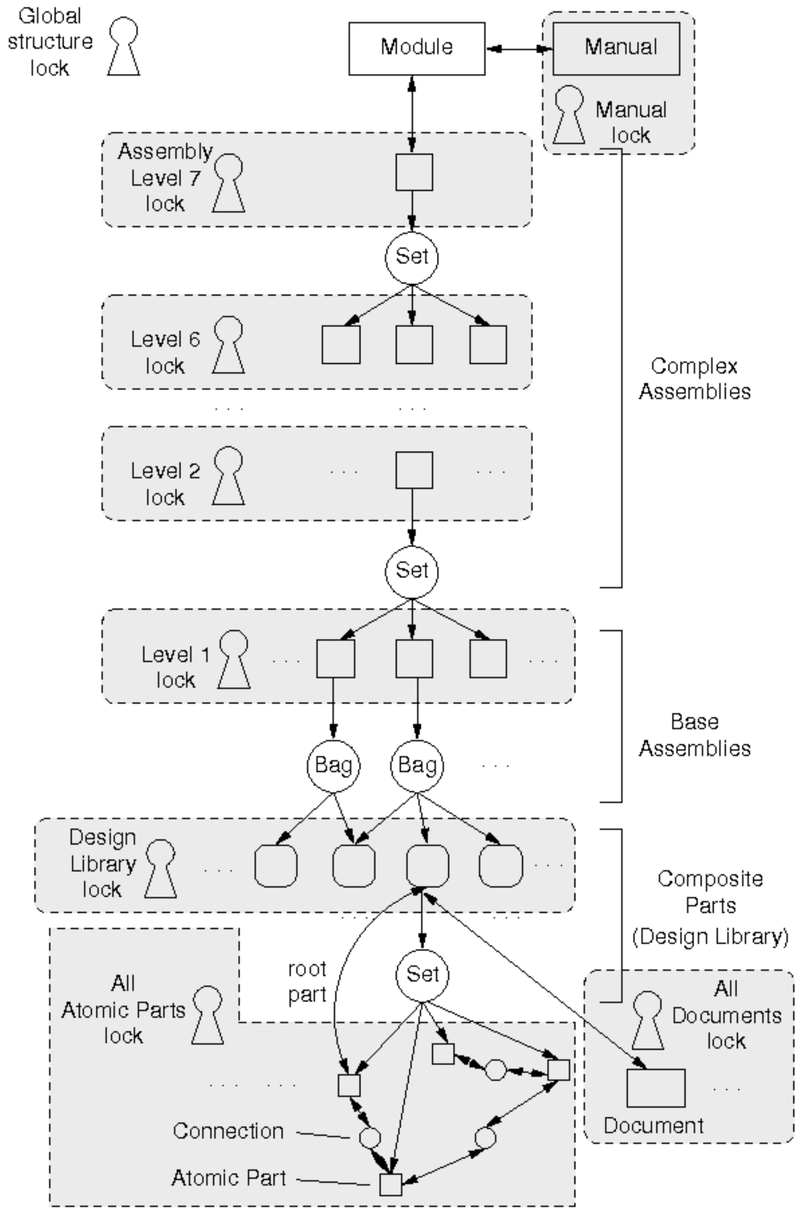
- New label of I is:

$$(\{A, C, H\} \cap \{A, C, G\} \cap \{A, C, G, J\}) \cup \{I\} = \{A, C, I\}$$

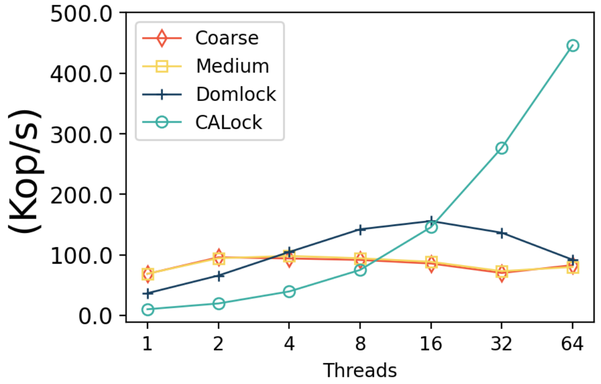


Performance - STMBench7

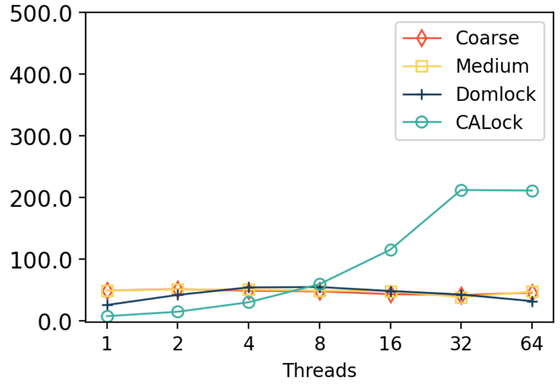
- What is the throughput of CALock compared other lock strategies
- How long does a thread spend waiting for a lock
- What is the cost of maintaining metadata for CALock and DomLock
- What is the relative sizes of lock grains between DomLock and CALock



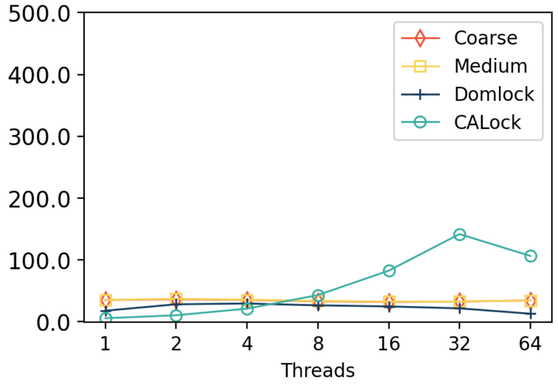
Throughput of locking strategies



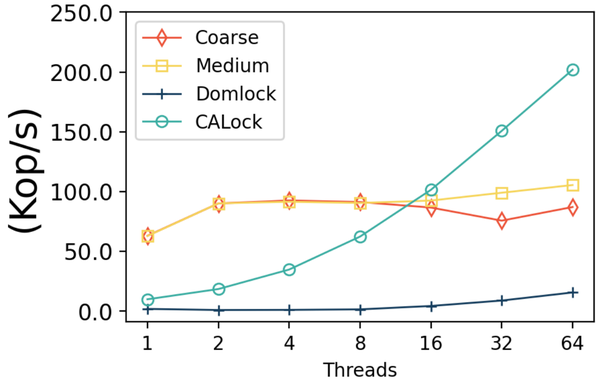
R:90% W:10%



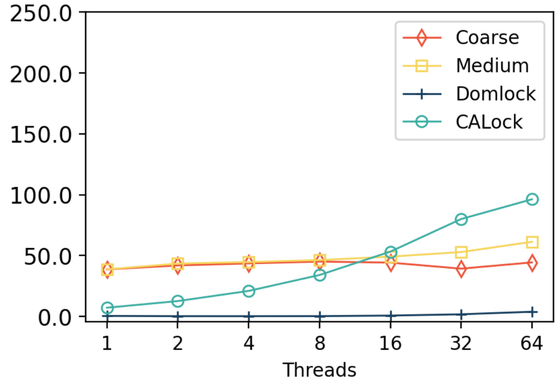
R:60% W:40%



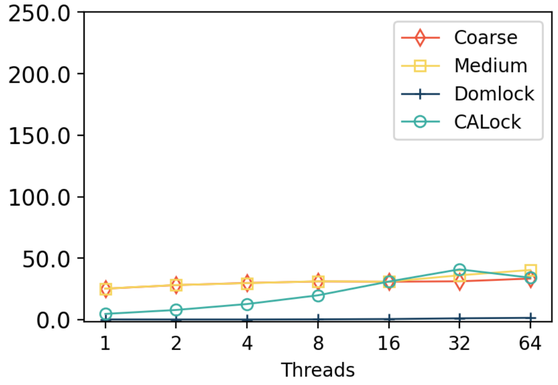
R:10% W:90%



R:90% W:9.9% M:0.1%

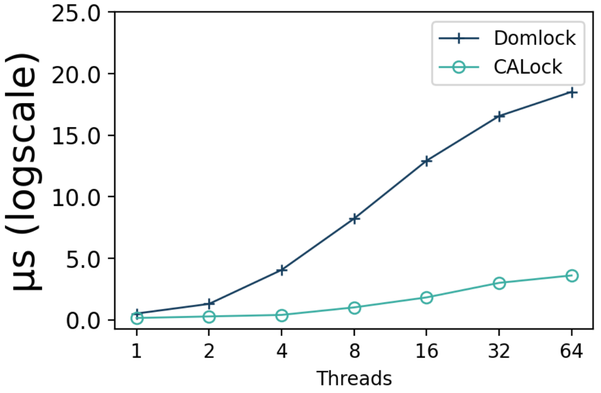


R:60% W:39.6% M:0.4%

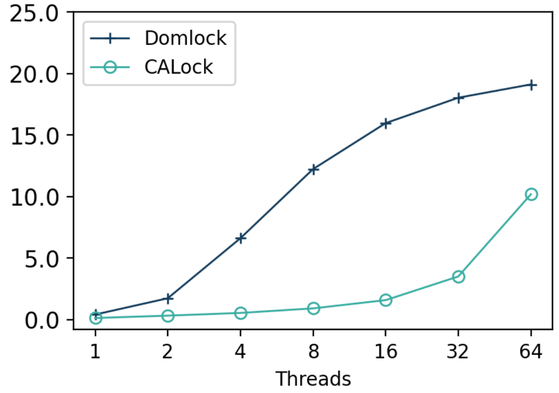


R:10% W:89.1% M: 0.9%

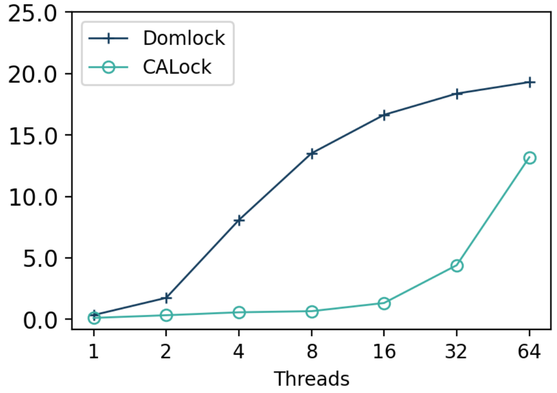
Response time for locks with CALock vs DomLock



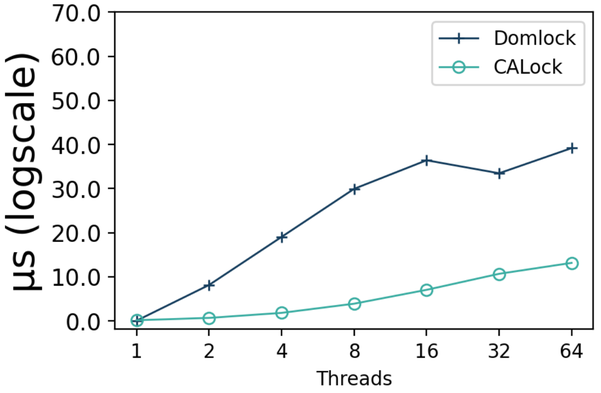
R:90% W:10%



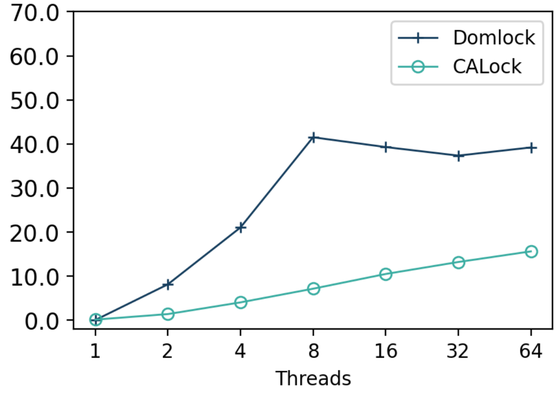
R:60% W:40%



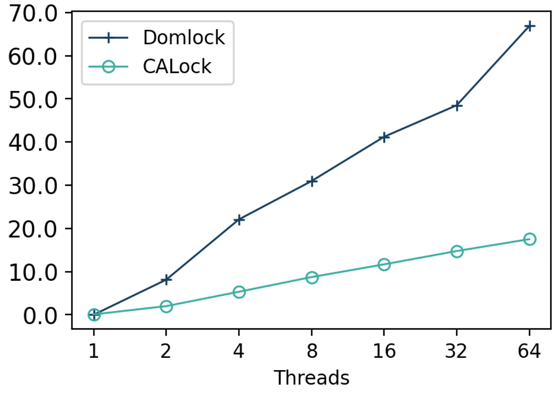
R:10% W:90%



R:90% W:9.9% M:0.1%

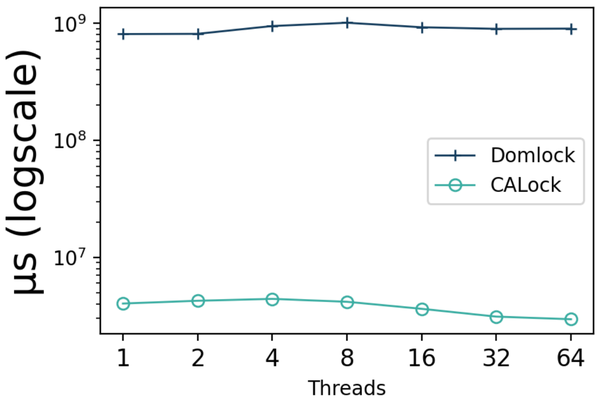


R:60% W:39.6% M:0.4%

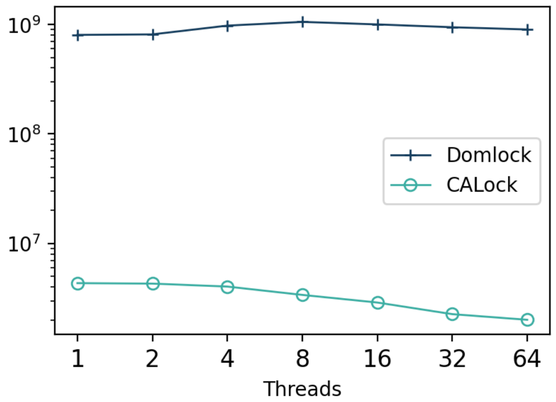


R:10% W:89.1% M: 0.9%

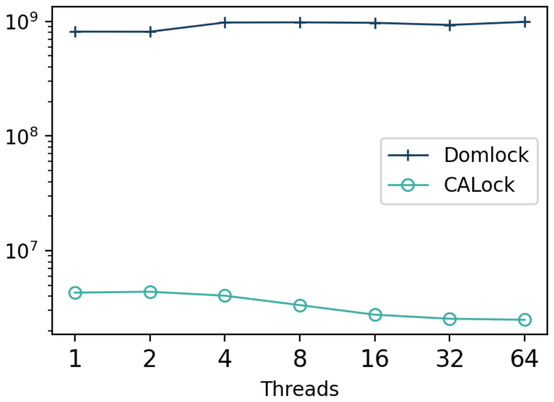
Cost of maintaining labels



R:90% W:9.9% M:0.1%

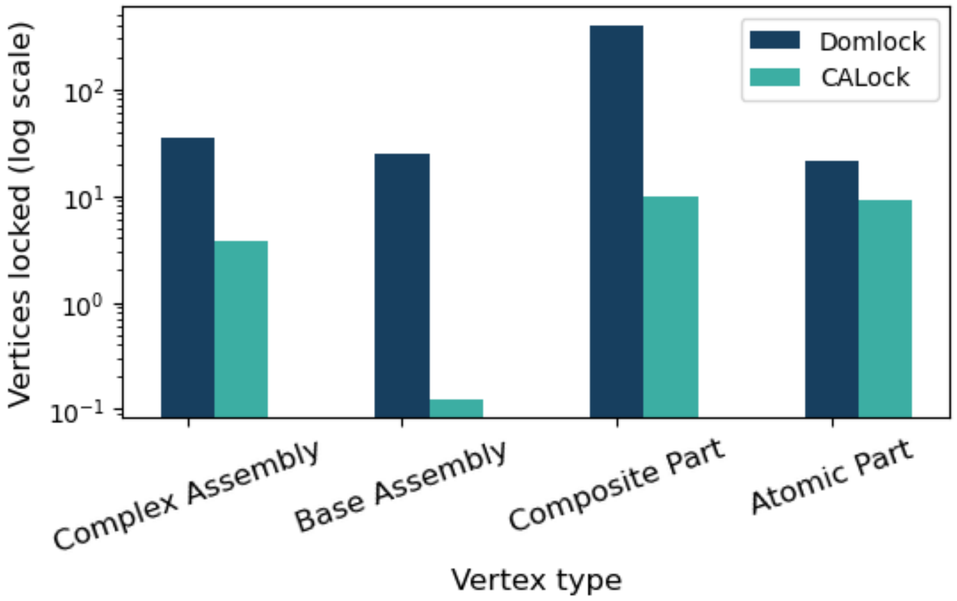


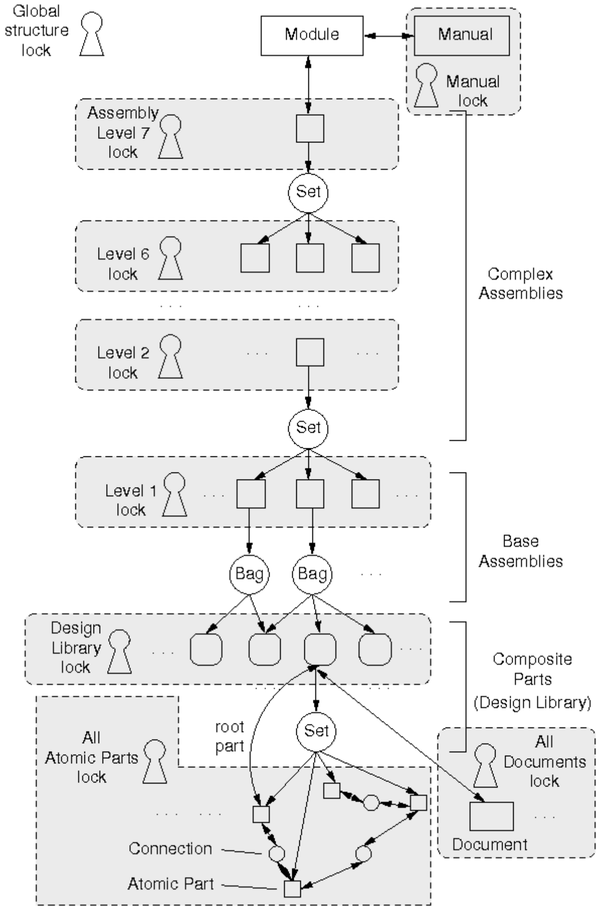
R:60% W:39.6% M:0.4%



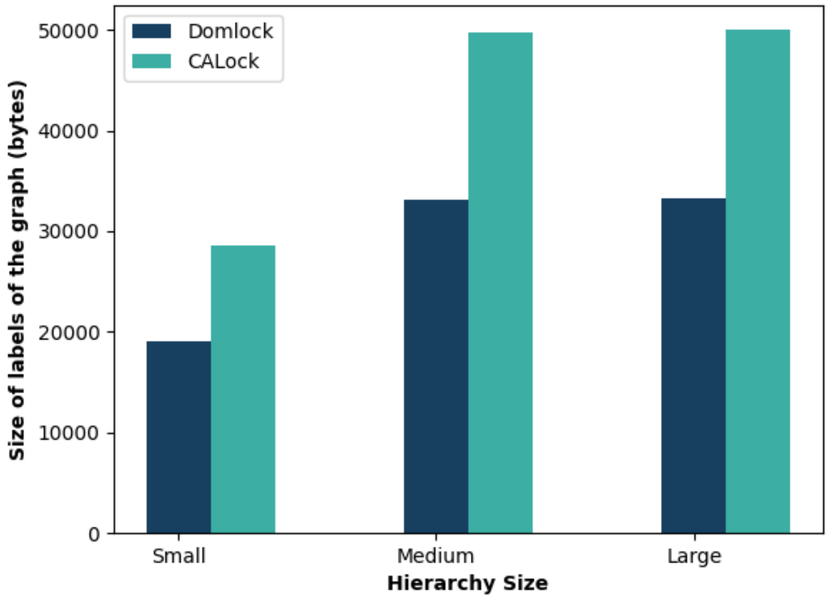
R:10% W:89.1% M: 0.9%

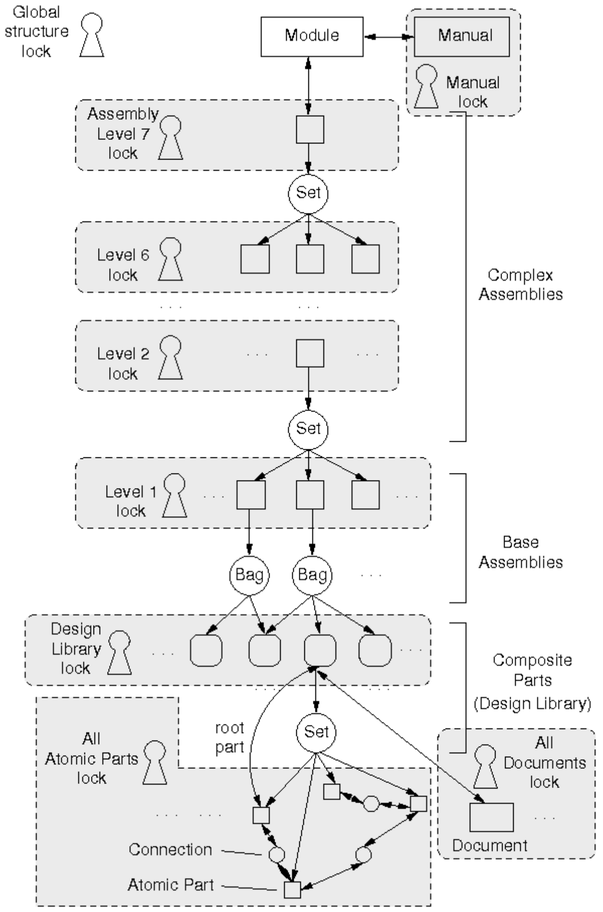
Grain sizes with Domlock and CALock





Size of labels in memory with Domlock and CALock





Outlook

- Allowing threads to hold multiple locks
- Allowing lock grains to be resized
- Eliminating the constraint that the graph is rooted

Merci !

Questions?