Unleashing Concurrency for Irregular Data Structures

By Peng Liu and Charles Zhang

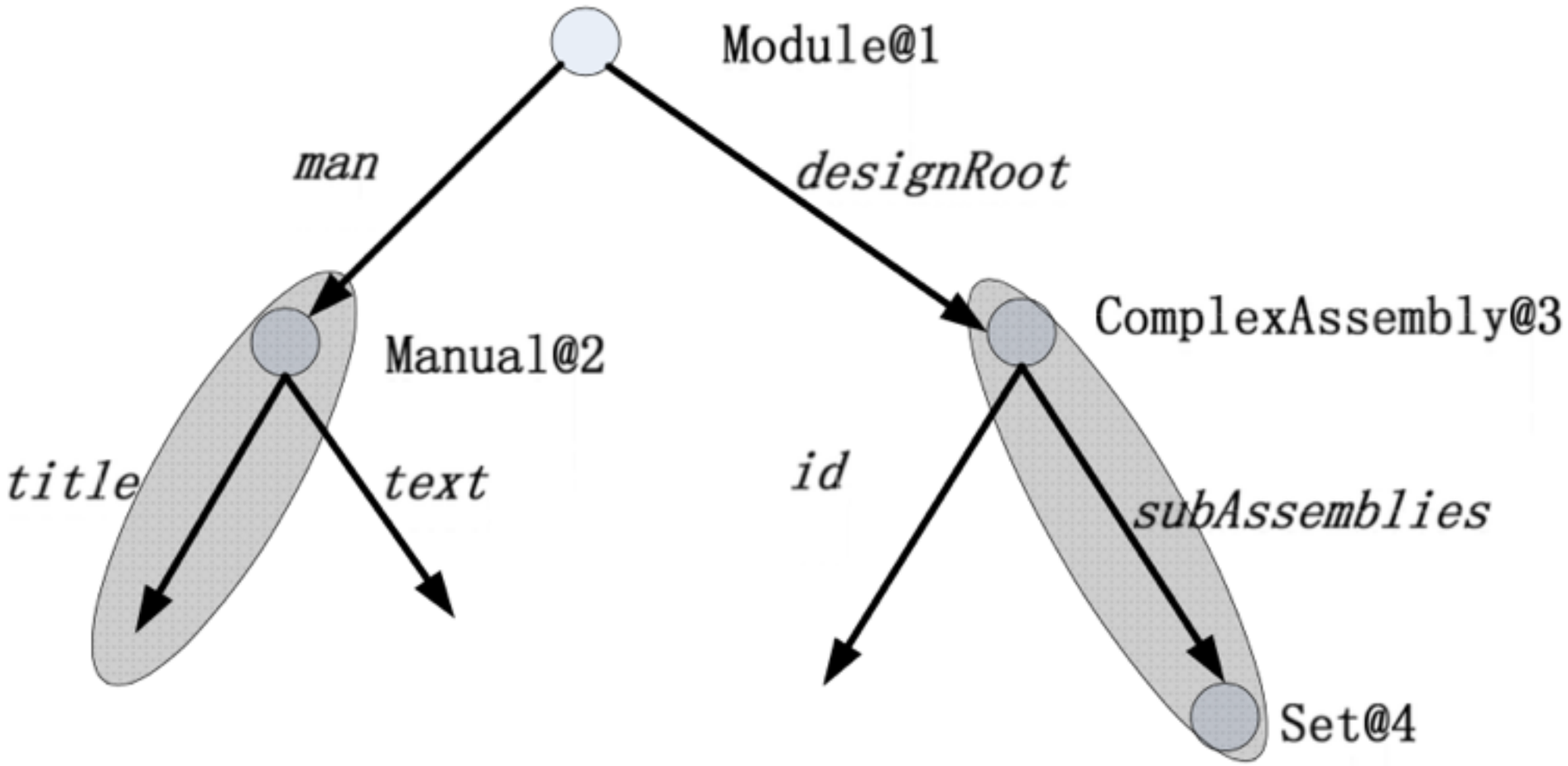
CCC Seminar

Álvaro Marco

Introduction

- In irregular data structures (graphs, sets…)
  - Usually coarse-grained locking — low degree of concurrency
  - Try to convert coarse-grained into fine-grained locking
- The method is applied to libraries

Motivational example



Different kinds of locks

- Multiple Granularity Locks (MGL)
  - X lock, applied when the field *f* is updated
  - IX lock, intention lock for X
  - S lock, applied when the field *f* is read
  - IS lock, intention lock for S

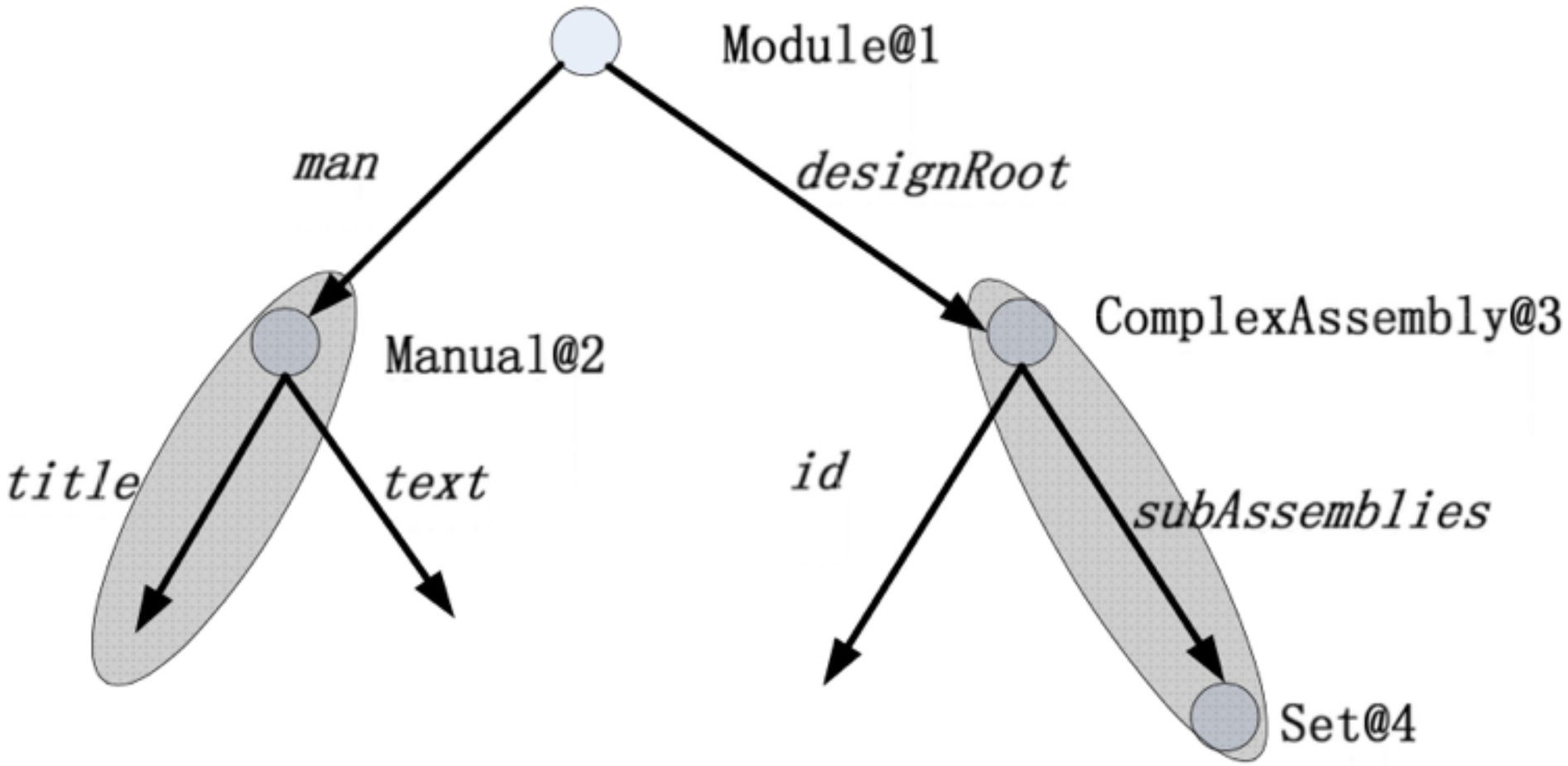
Compatibility of locks

	IS	IX	X	S
IS	✓	✓	x	✓
IX	✓	✓	x	x
X	x	x	x	x
S	✓	x	x	✓

Technique summary

1. Construct the Abstract Object Graph (AOG)
2. Apply MGL tags depending on the effects of the AOG
3. Given the tagged AOG produce the optimized locking

1. Construct the AOG



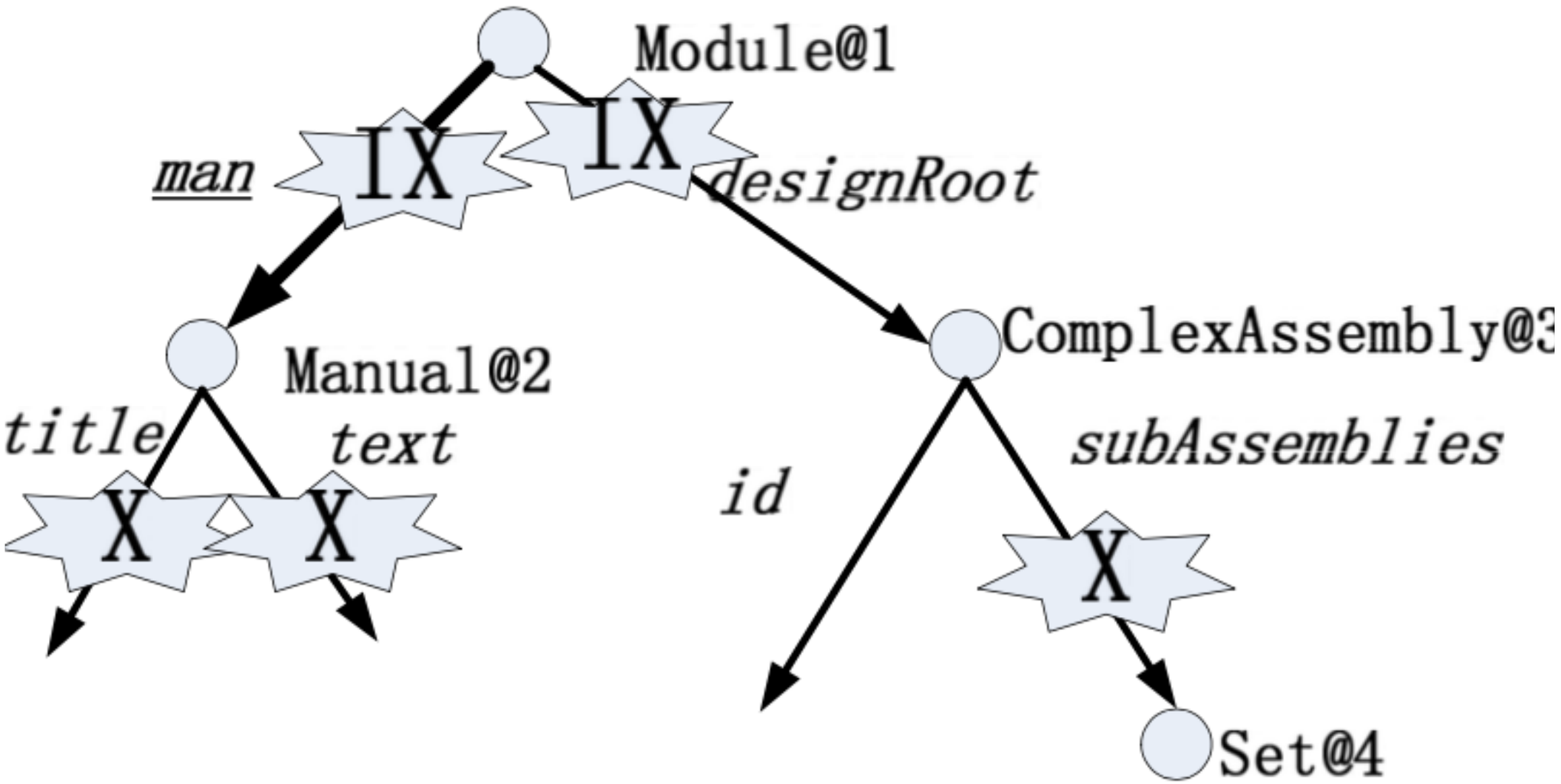
2. Tag the AOG

- Tag each field based on how the field is accessed in the atomic method
- Example of tagging rule: if a field is updated the edge that represents it is tagged with the X lock, and each edge in the path to the root is tagged with the IX lock

3. Produce the optimized lockings

- Reduce the number of tags in the tagged AOG
- Replace original synchronization
- *Deadlock-freeness* by fixing orders (e.g. topological order)

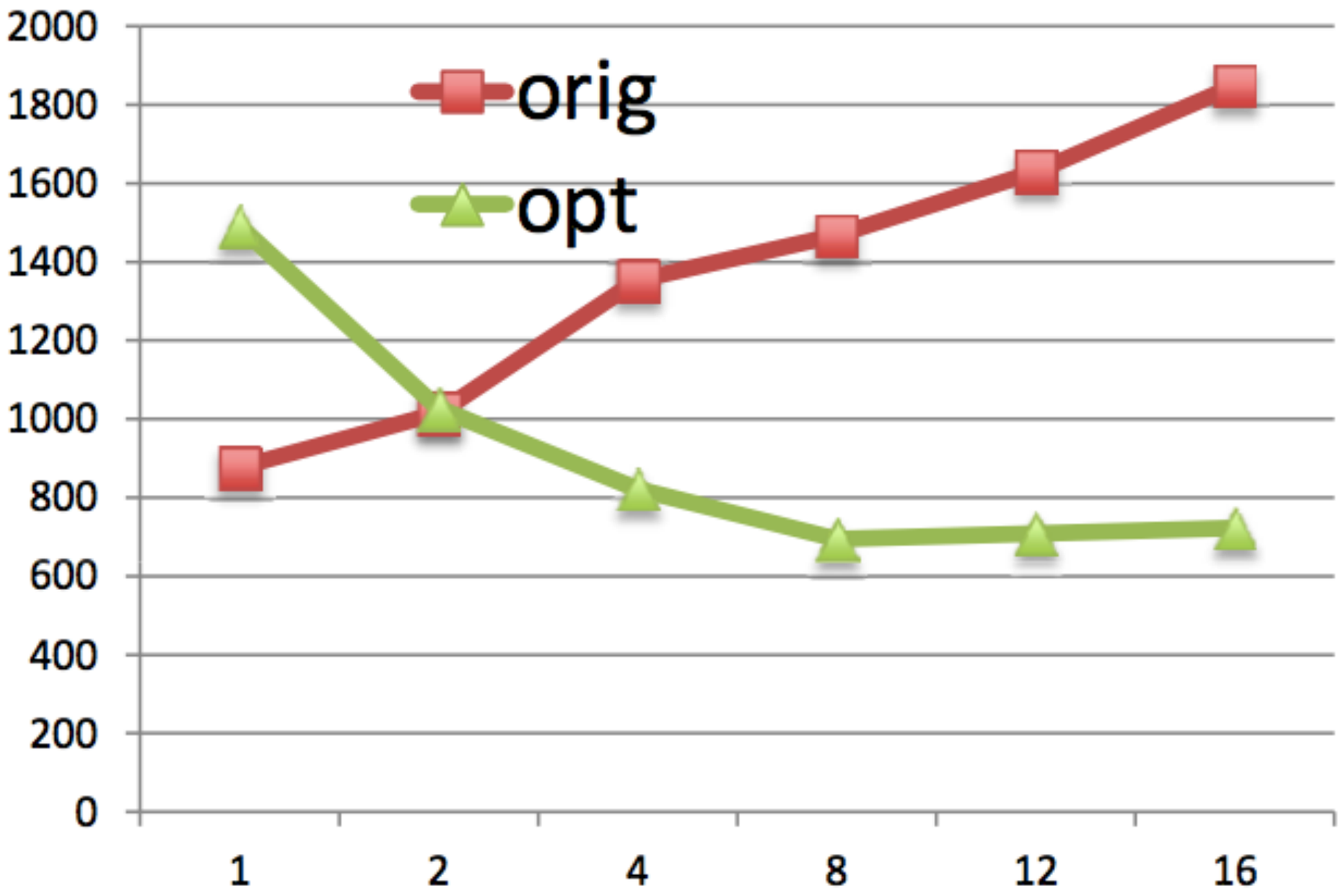
Tagging example



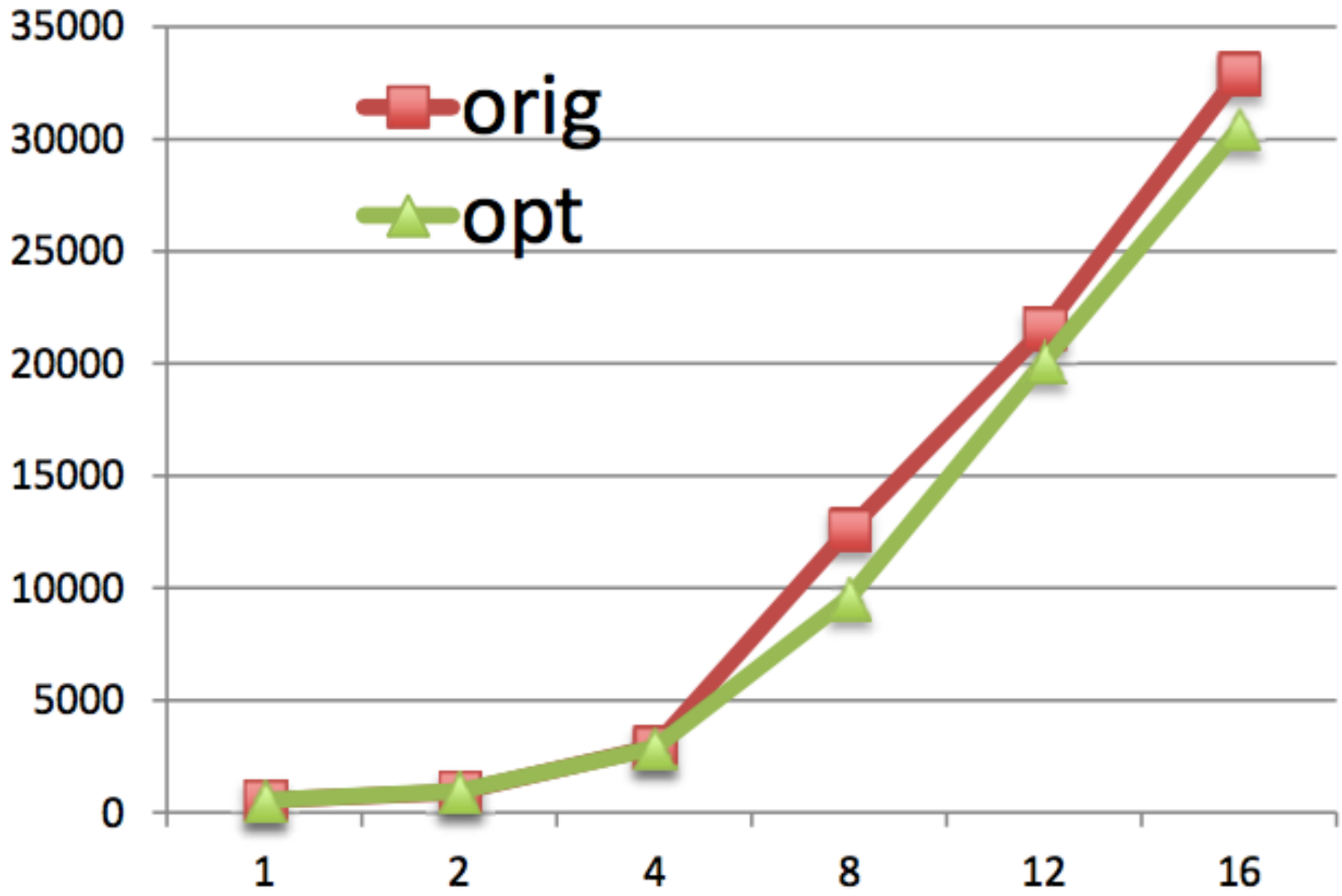
Collections

- The collections Map and Set are heavily used in real world applications
- Replace them by their concurrent versions
- When the key is available apply key-based locking otherwise coarse locking

Jgrapht performance



Tuplesoup performance



Conclusion

- The presented technique allows a better usage of resources
  - 7% to 2X speedup
- Speedup only when the number of threads is at least 4