school of | Computer Science & Statistics

An Experimental Comparison of Concurrent Data Structures

Student: Mark Gibson **Supervisor:** Dr. David Gregg

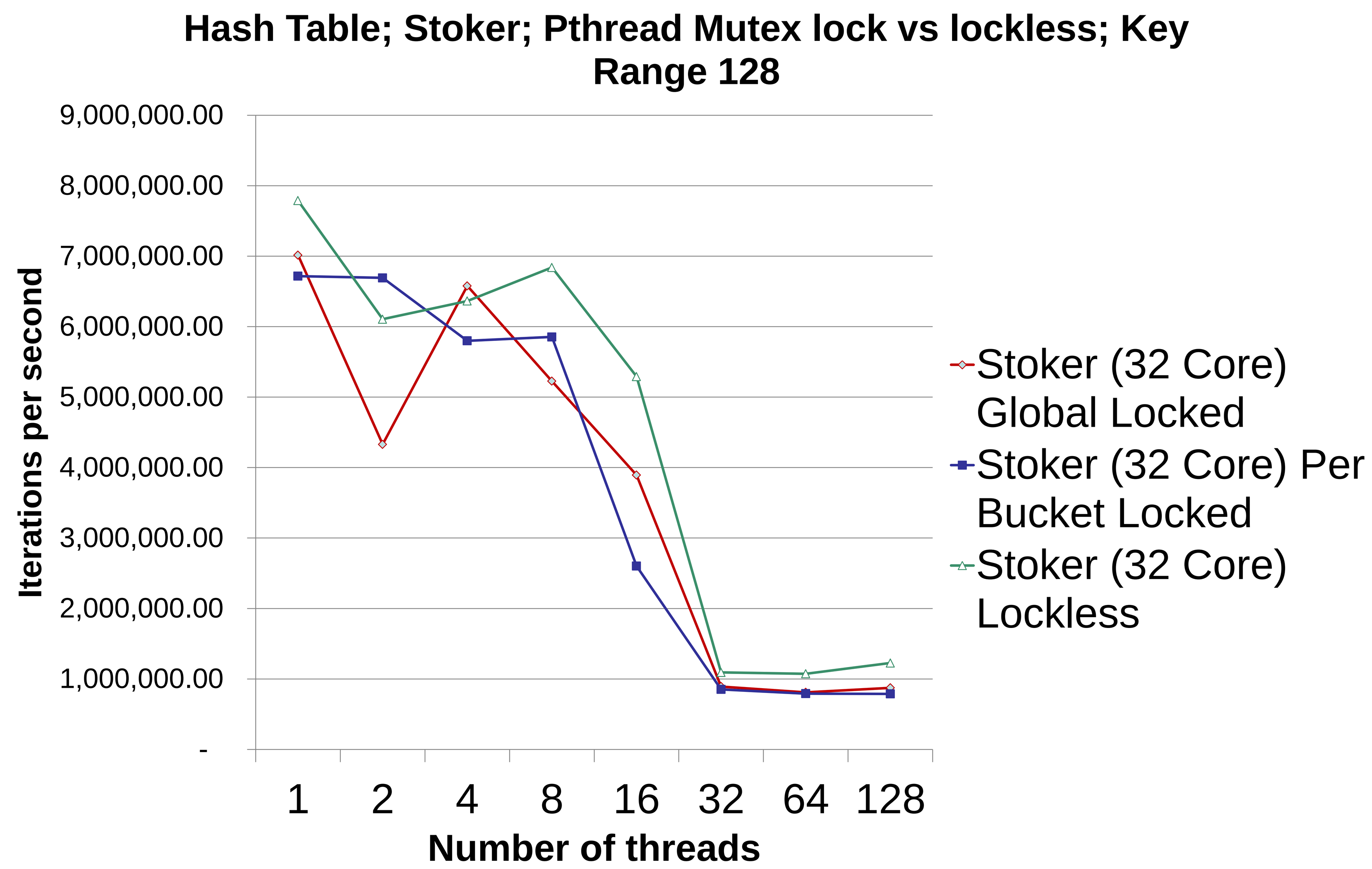
Background

- Concurrent Data Structures
  - Designed for access by multiple threads
  - Potential for high scalability
- Atomic Instructions
  - Either complete fully or not at all
  - Used to implement locked & lockless algorithms
- Locks
  - Use mutexes to acquire/release a lock
  - Blocks threads that don't have the lock
- Lockless
  - Use atomic instructions
  - Guaranteed system throughput

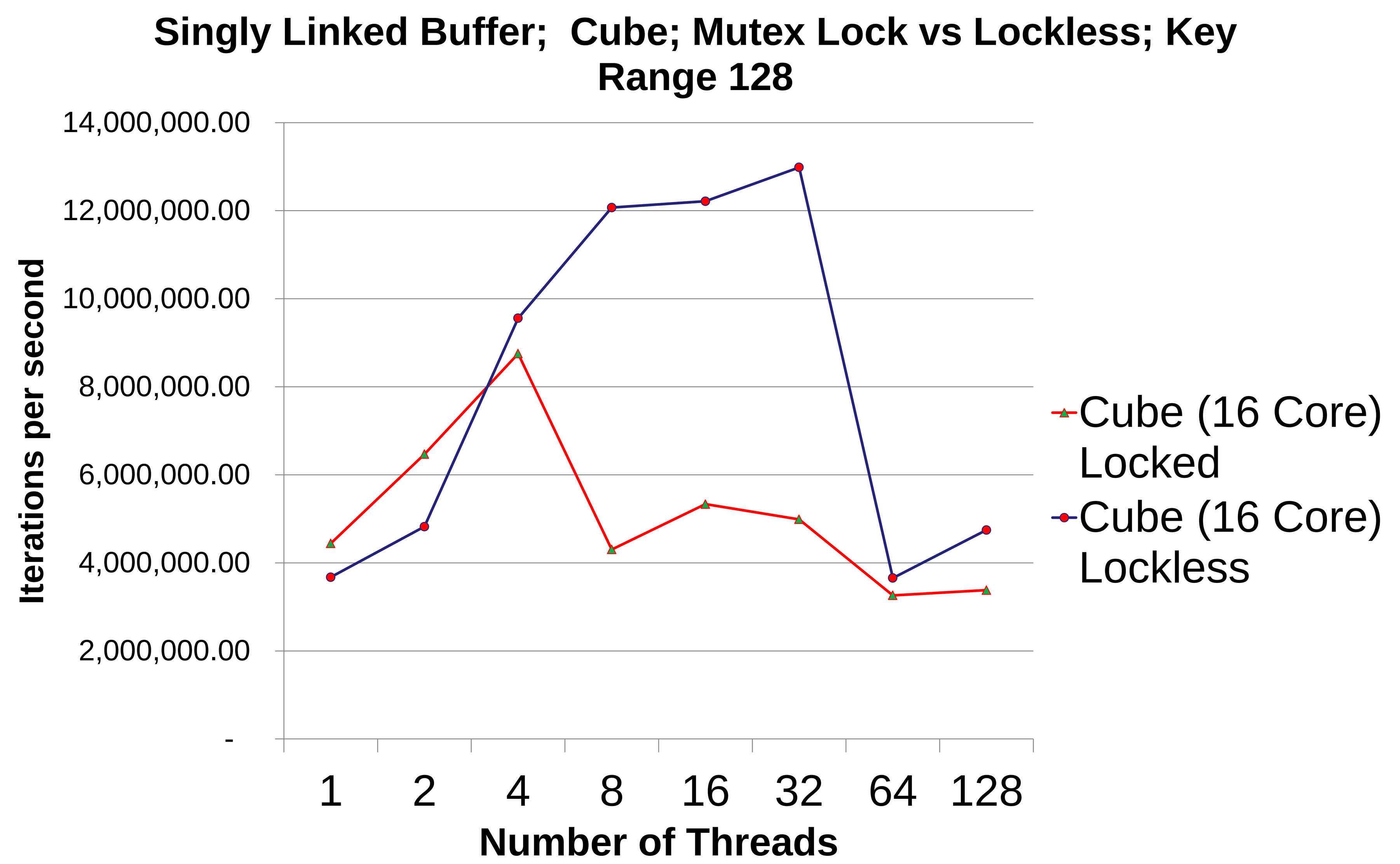
Approach

- Implemented three concurrent data structures
  - Ring Buffer
  - Linked List
  - Hash Table
- Lock Variations
  - Pthread Mutex
  - Test-and-Set (3 Variations)
  - Ticket (2 Variations)
- Lockless Variations
  - Compare-and-Swap
  - Fetch-and-Add
- 13 Variations for each Data Structure
  - 12 Locks, 1 Lockless
  - Compared each on 3 systems

Evaluation



	Global Lock	Bucket Lock	Lockless
CPU Cycles	1.13x10^12	9.86x10^11	1.09x10^12
Stalled Front End	1.06x10^12	9.16x10^11	1.02x10^12
Stalled Back End	6.89x10^11	5.43x10^11	6.27x10^11



	Cube Lockless	Cube Locked
CPU Cycles	1.66x10^10	3.70x10^10
Stalled Front End	8.06x10^9	2.81x10^10
Stalled Back End	4.29x10^9	1.92x10^10

Conclusion

- The project was successful in performing an in-depth comparison of 3 data structures
- Lockless algorithms were mostly faster
  - Some exceptions
- The linked list had the best performance gain
  - Using Lockless algorithm versus Lock
- The hash table had the worst performance
  - Outperformed by the locked variations

BA (Mod) in Computer Science

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH
THE UNIVERSITY OF DUBLIN

Mark Gibson - gibsonma@tcd.ie