Optimizing Write-Heavy Database Operations Using B$^\varepsilon$-Trees

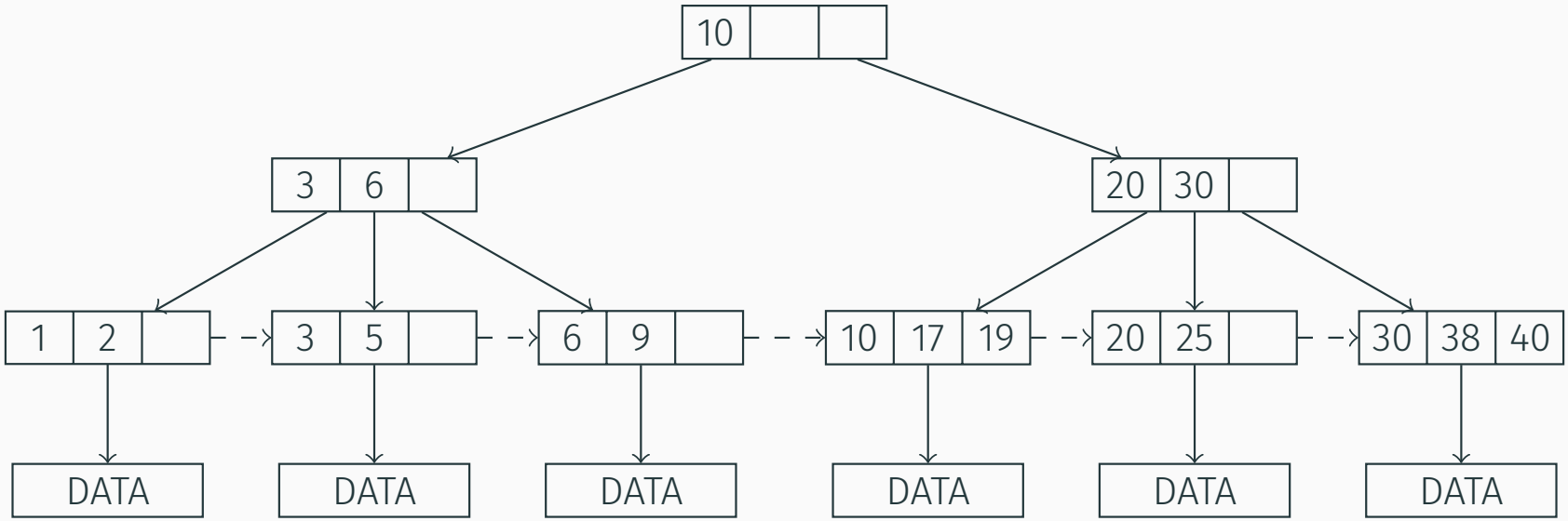
Christoph Rotte

September 1, 2022

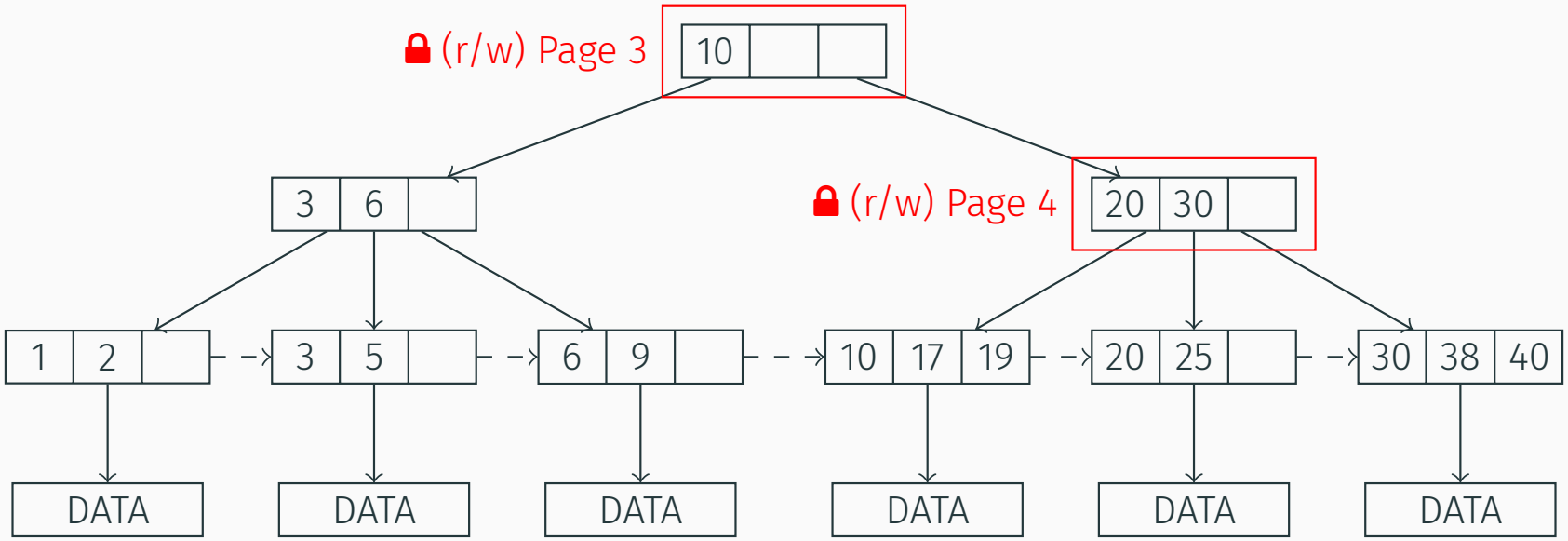
Bachelor's Thesis - Final Presentation

Chair for Database Systems | Technical University of Munich

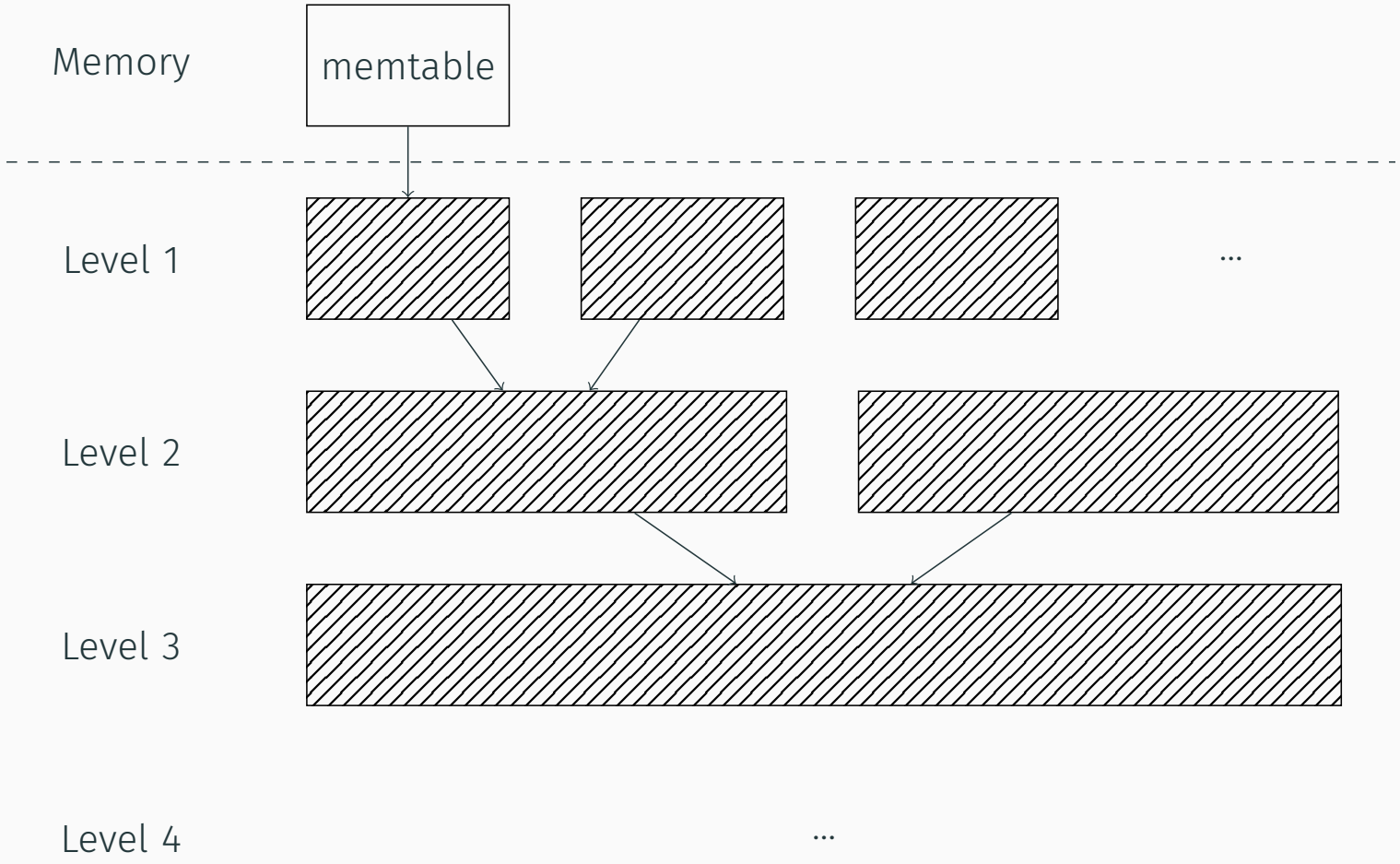
B/B$^+$-Trees in Databases



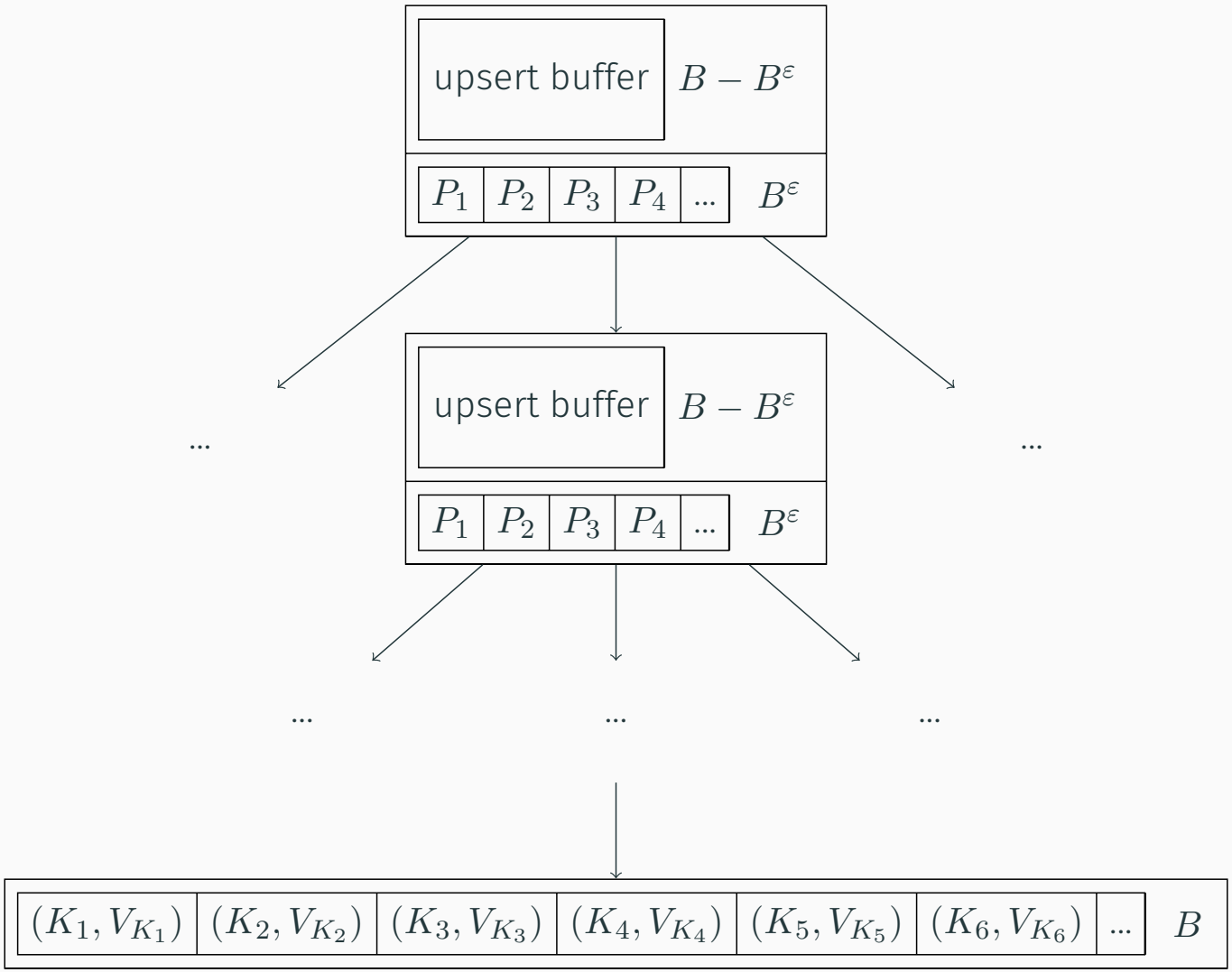
B/B$^+$-Trees in Databases



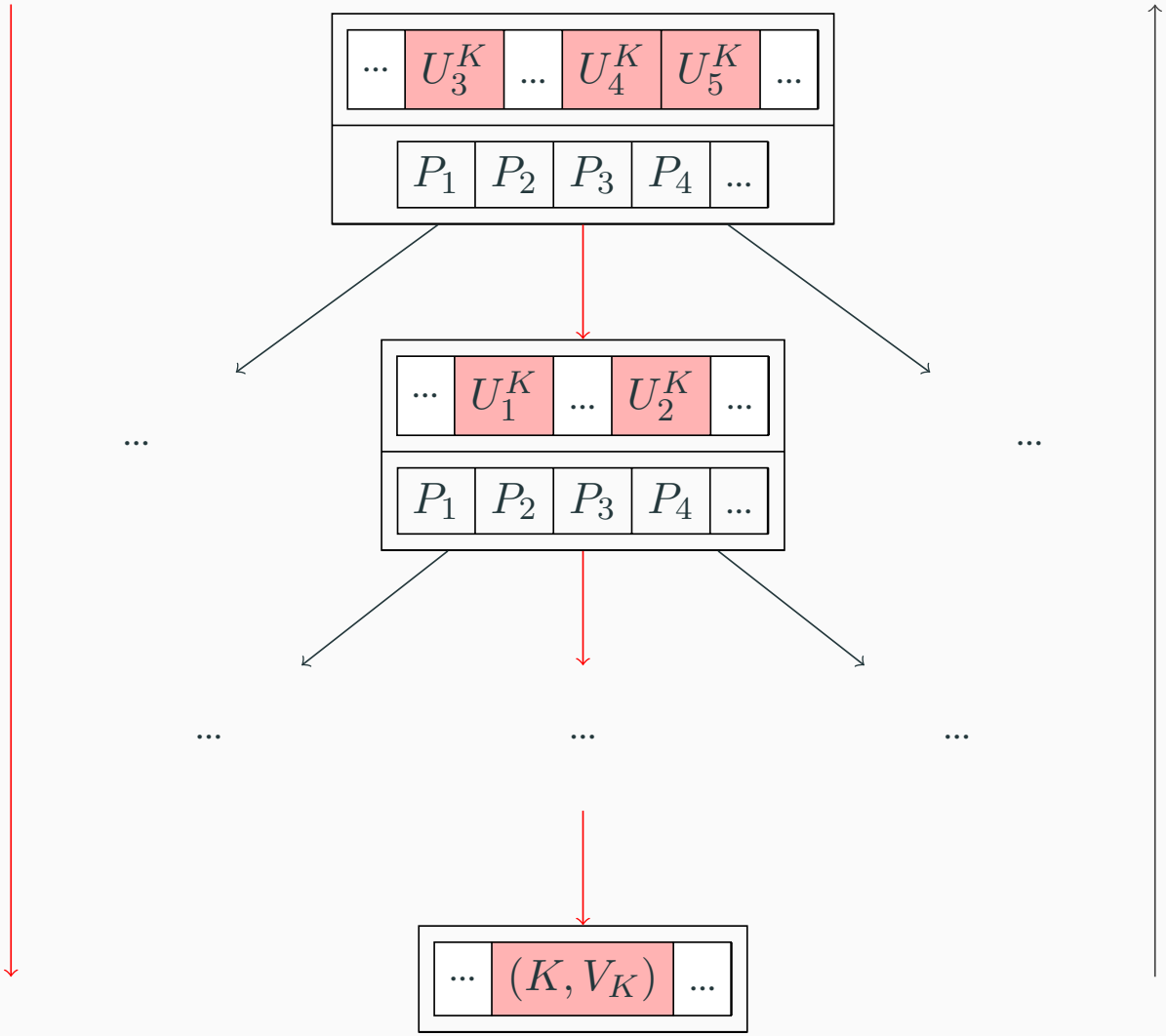
Log-Structured Merge-Trees (LSM-Trees)



B^ε-Trees | Structure



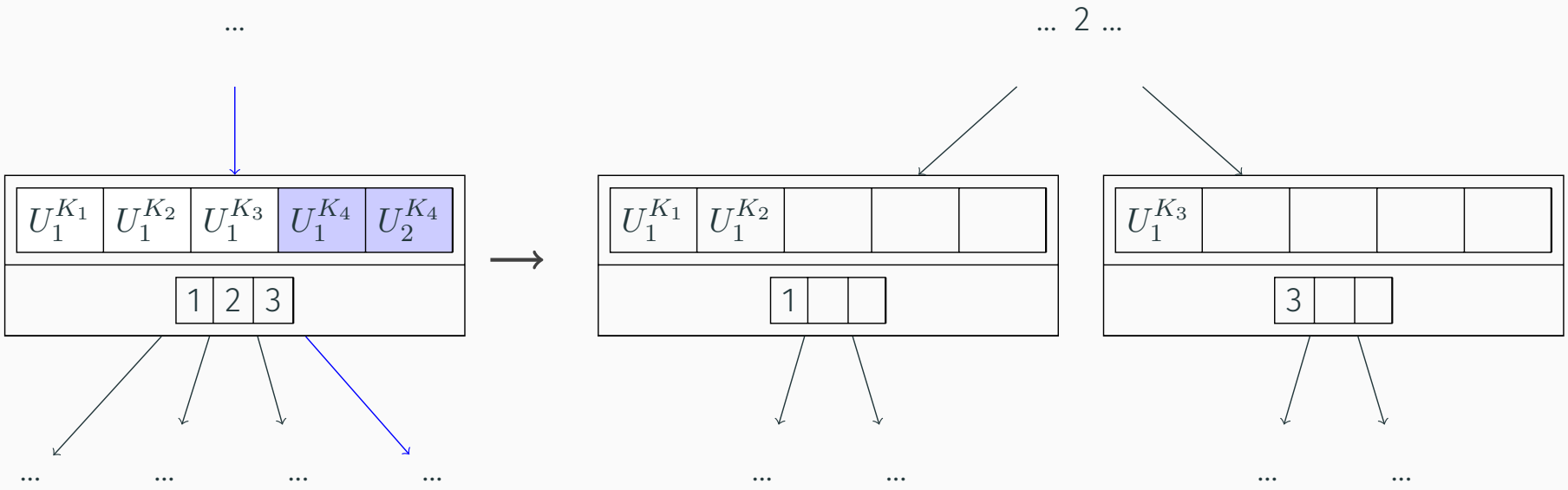
B^{ε}-Trees | Lookups



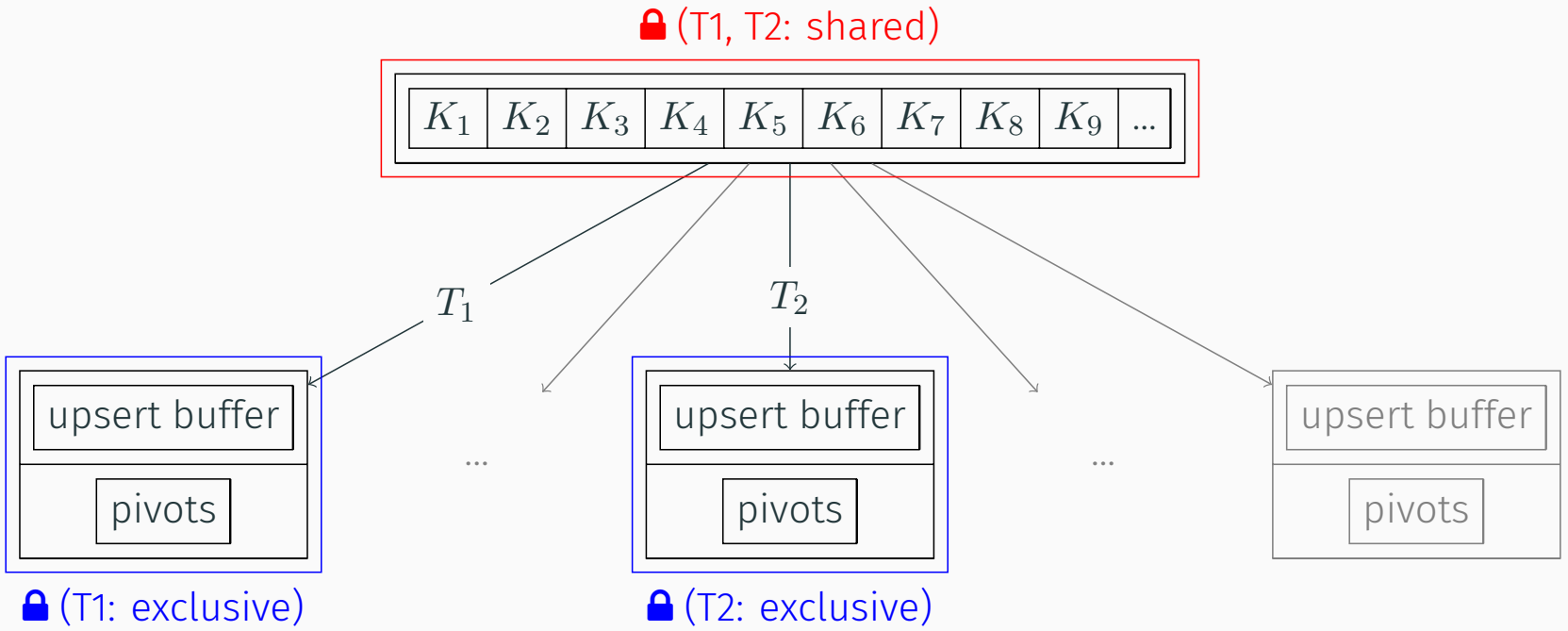
Asymptotic Comparison | I/O Operations

	B/B$^+$-Tree	B$^\varepsilon$-Tree	LSM-Tree
Upserts	−	+	+
Lookups	+	+	−

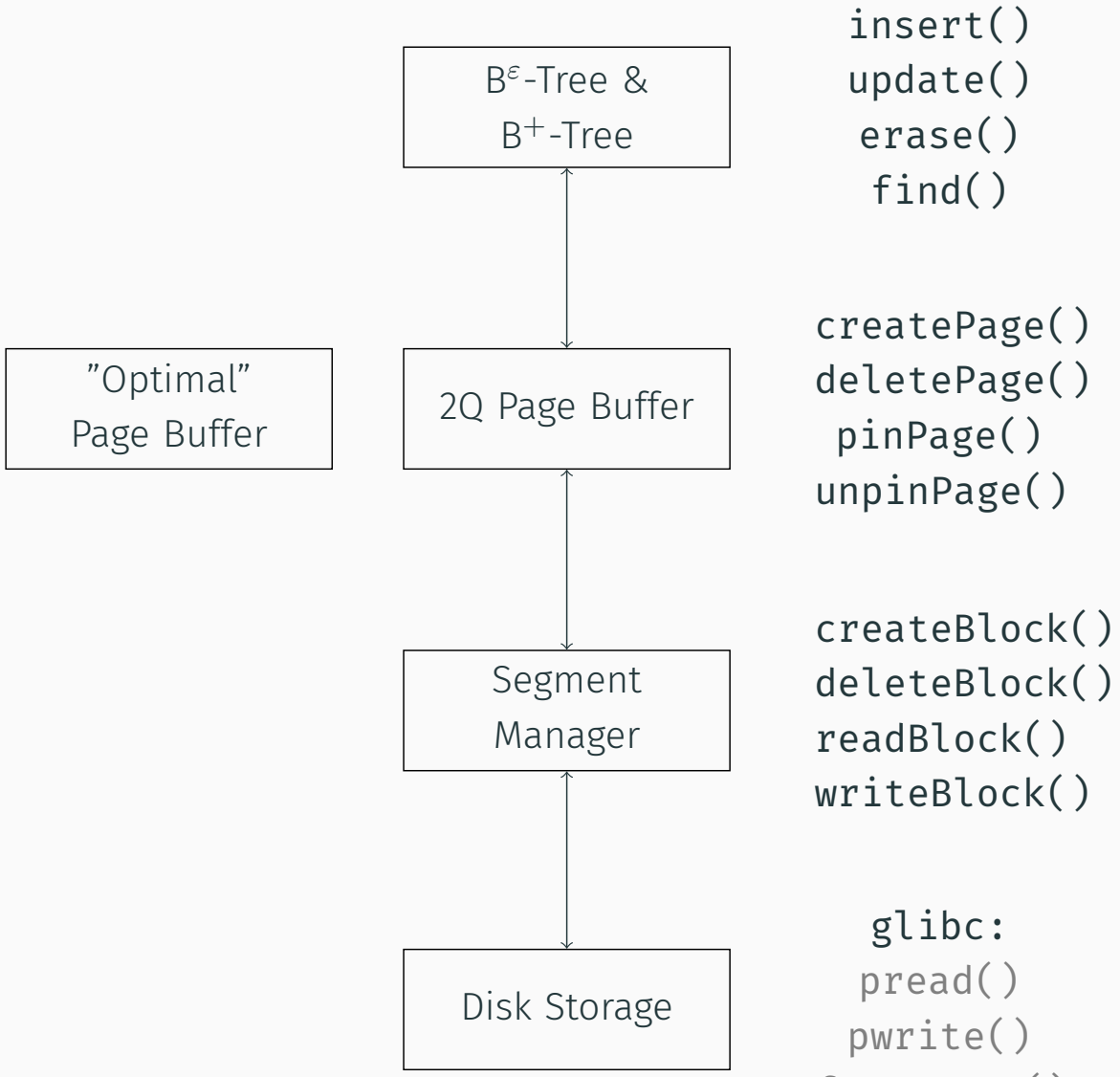
Implementation | Preemptive Splitting



Implementation | Separate Root Node



Implementation | Design Layers

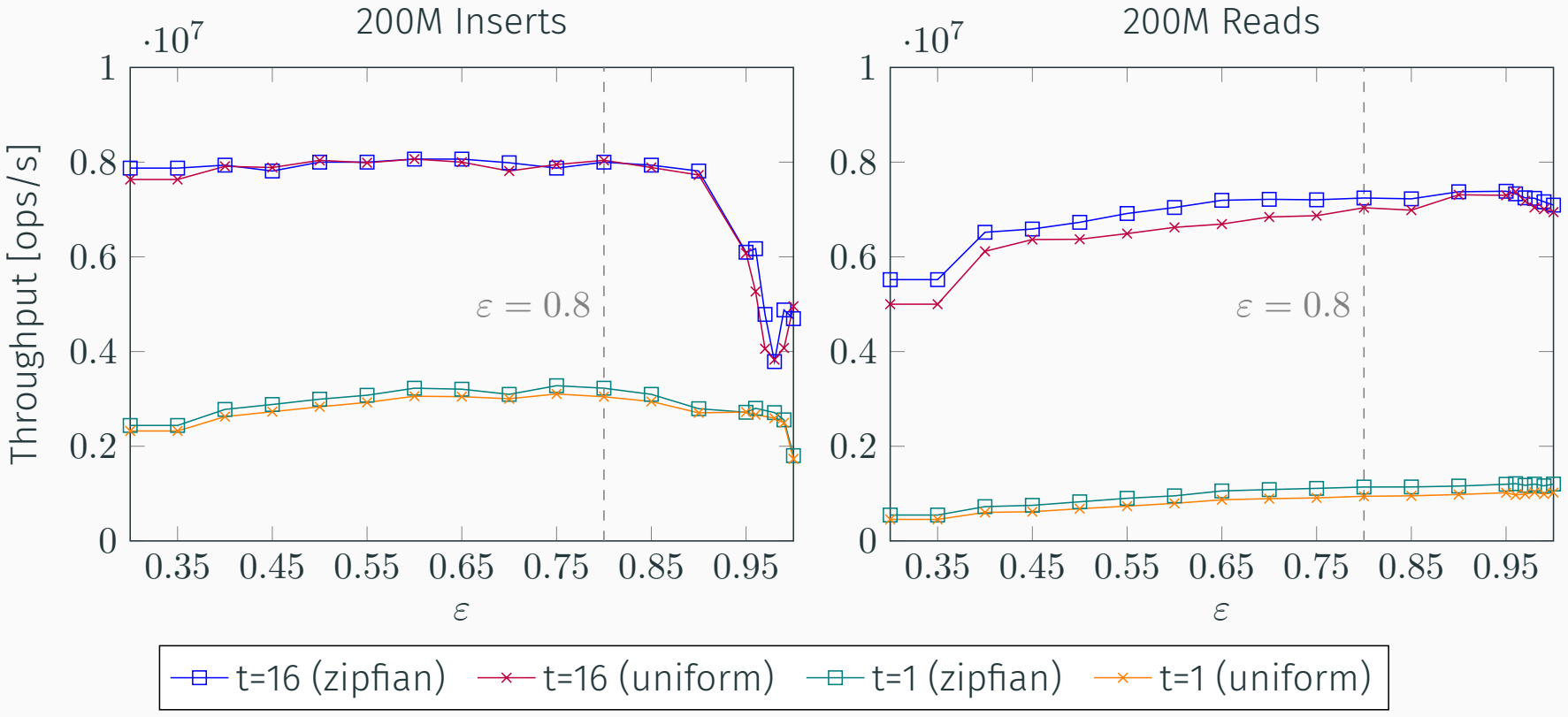


Evaluation | Benchmark Setup

- **Benchmark Tool**: Unum Cloud Benchmark (UCSB)
  $\rightarrow$ rewrite of YCSB in C++
- Intel i9-7900X (20 threads) | Samsung SSD 970 EVO
- **Sizes**: 16KiB pages | 8B keys | 100B values
- **Distribution**: Zipfian (constant: 0.99)

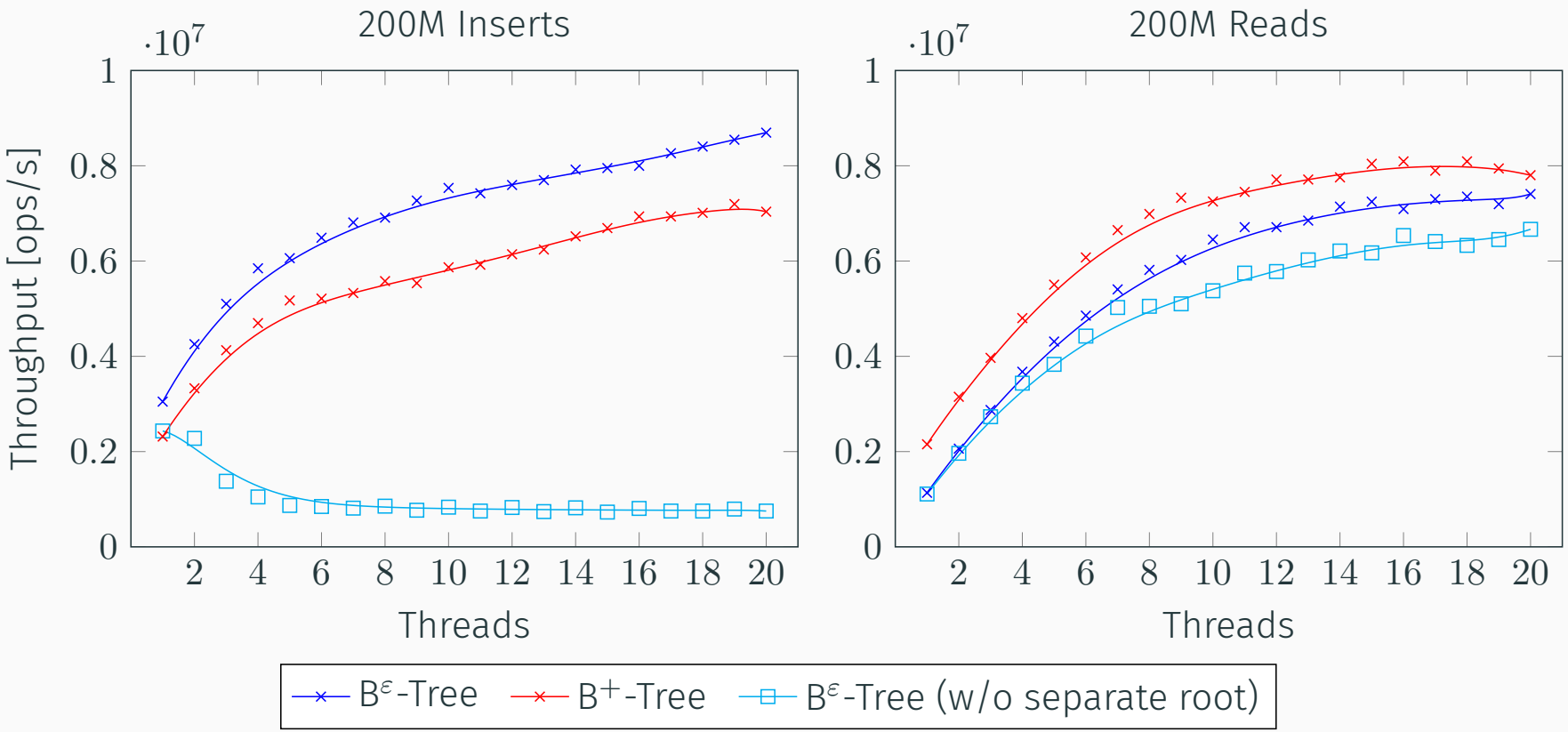
”Optimal” Buffer | 200M Operations w/ 200M Preloaded Values

Buffered Binary Tree ($\varepsilon = 0$) → B$^+$-Tree ($\varepsilon = 1$)

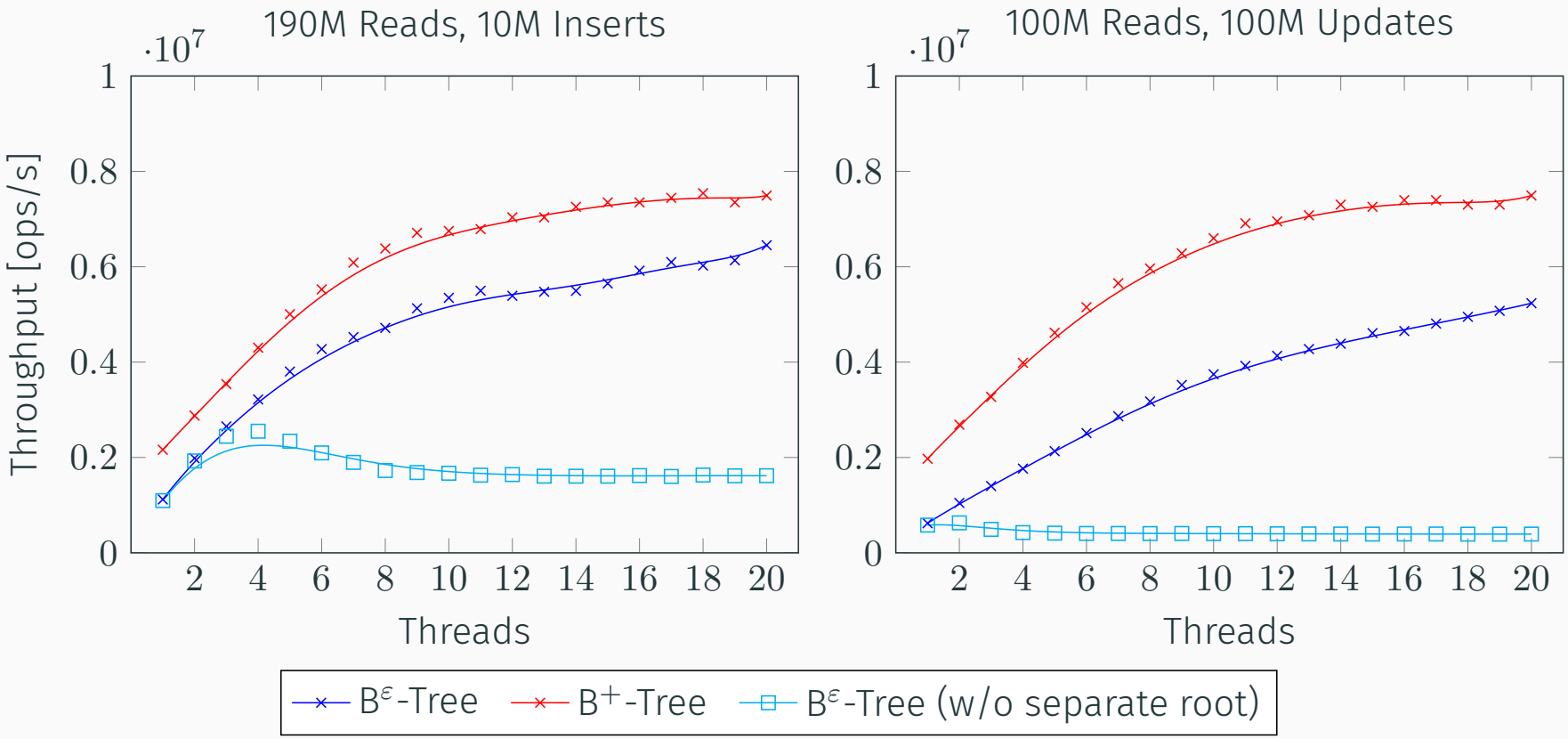


$\varepsilon = 0.8 \rightarrow$ buffer slots = 109, pivots = 147

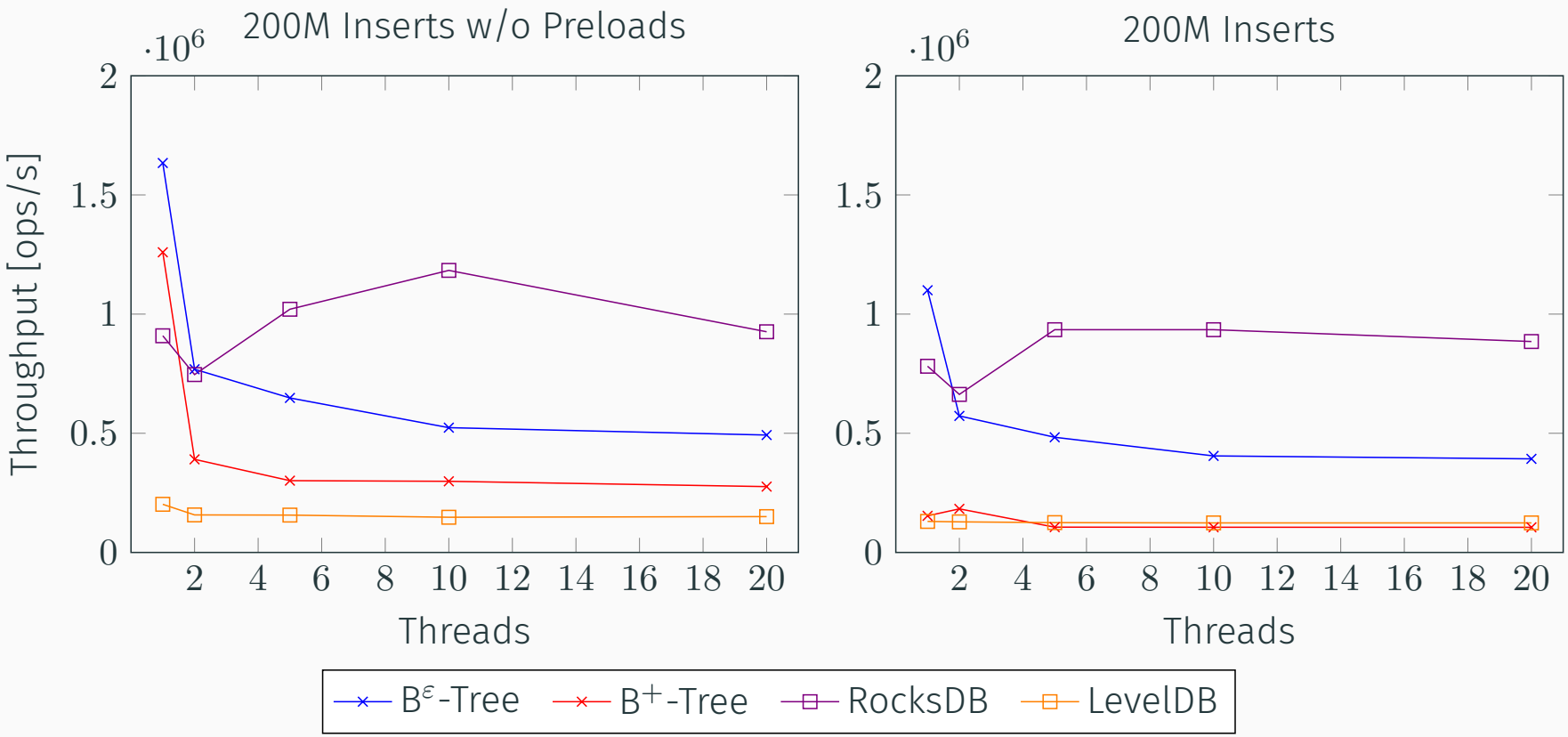
”Optimal” Buffer | 200M Operations w/ 200M Preloaded Values



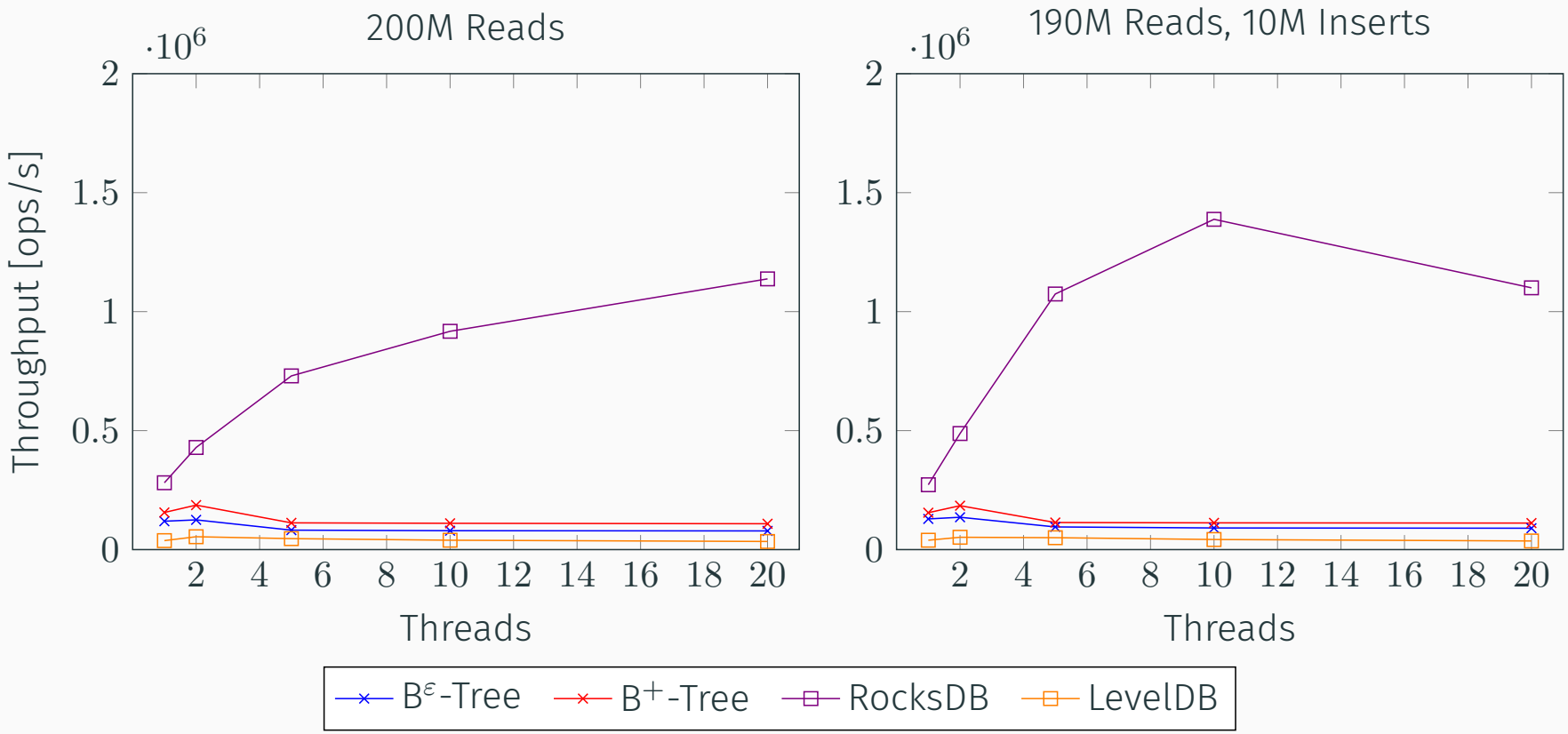
”Optimal” Buffer | 200M Operations w/ 200M Preloaded Values



2Q Buffer (10GB) | 200M Operations w/ 200M Preloaded Values



2Q Buffer (10GB) | 200M Operations w/ 200M Preloaded Values



B^ε-Trees | Conclusion

- Asymptotically between B/B^+-Trees and LSM-Trees
- Textbook B^ε-Trees require adaption for practical use
- **Future work**:
  - Advanced page buffer
  - Optimistic Lock Coupling (OLC)
  - ...