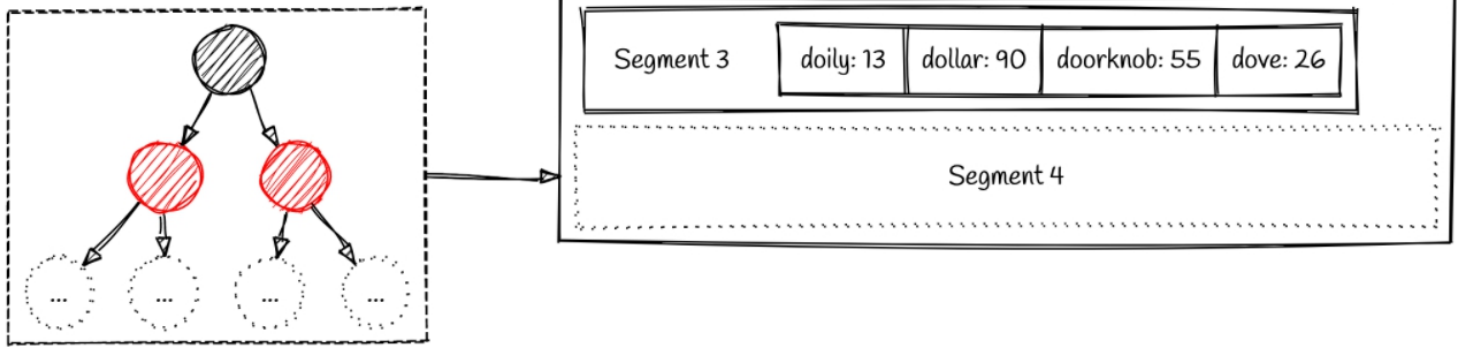
NovKV: Efficient Garbage Collection for Key-Value Separated LSM-Stores

Chen Shen, Youyou Lu*, Fei Li, Weidong Liu*, Jiwu Shu
Tsinghua University

slide made by wgl

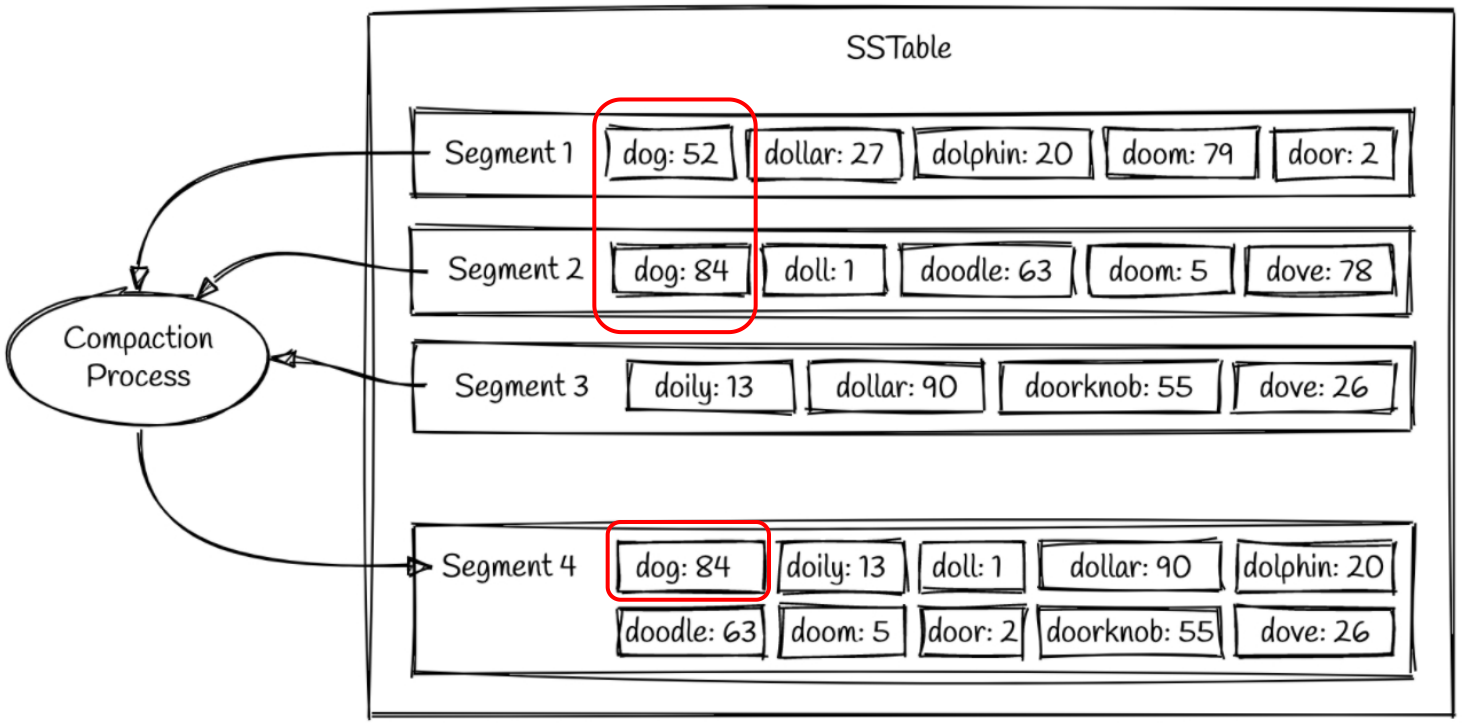
Background

- **LSM-tree Basics**
  - Log Structured Merge Tree
  - SSTable & Segment
- **Problem**
  - read amplification



Background

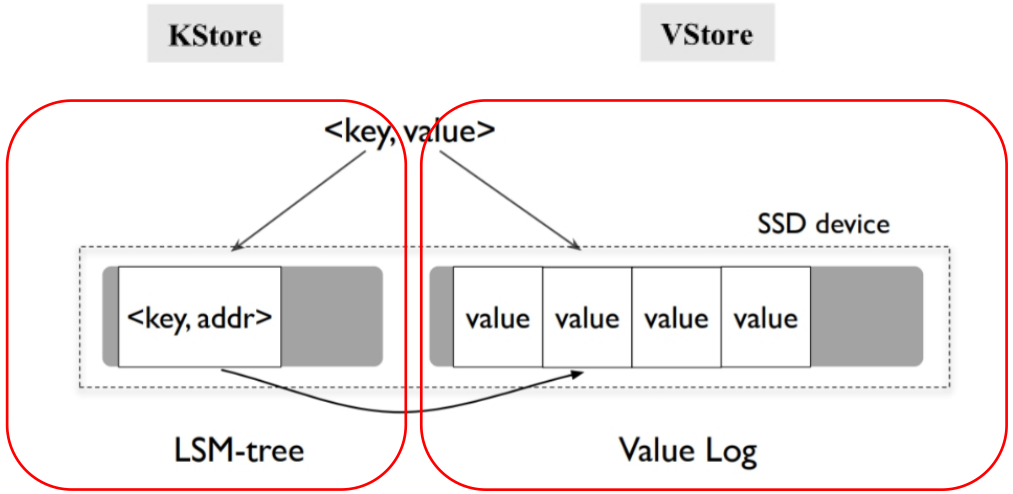
- **LSM-tree Compaction**
  - Overlap of key ranges
  - Update invalid key
  - Compaction & Overwrite
- **Problem**
  - write amplification
  - resource contention

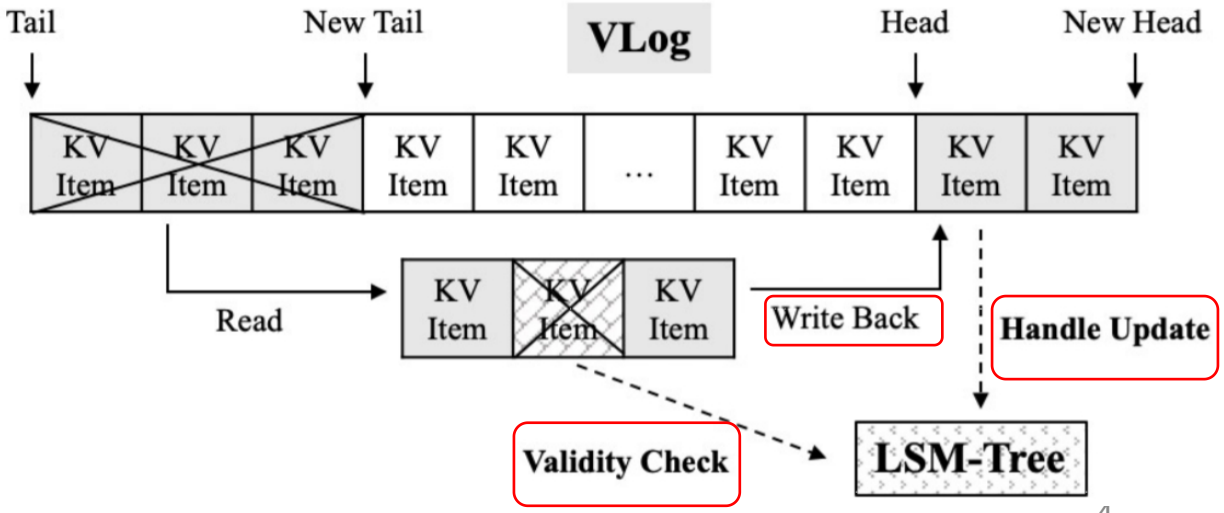


Motivation

- **LSM-tree Optimizations**
  - Key-Value separation
  - KStore->Key + Value Handle (epoch)
  - VStore->KVItem
- **Problem**
  - Garbage Collection
  - query and insert overheads

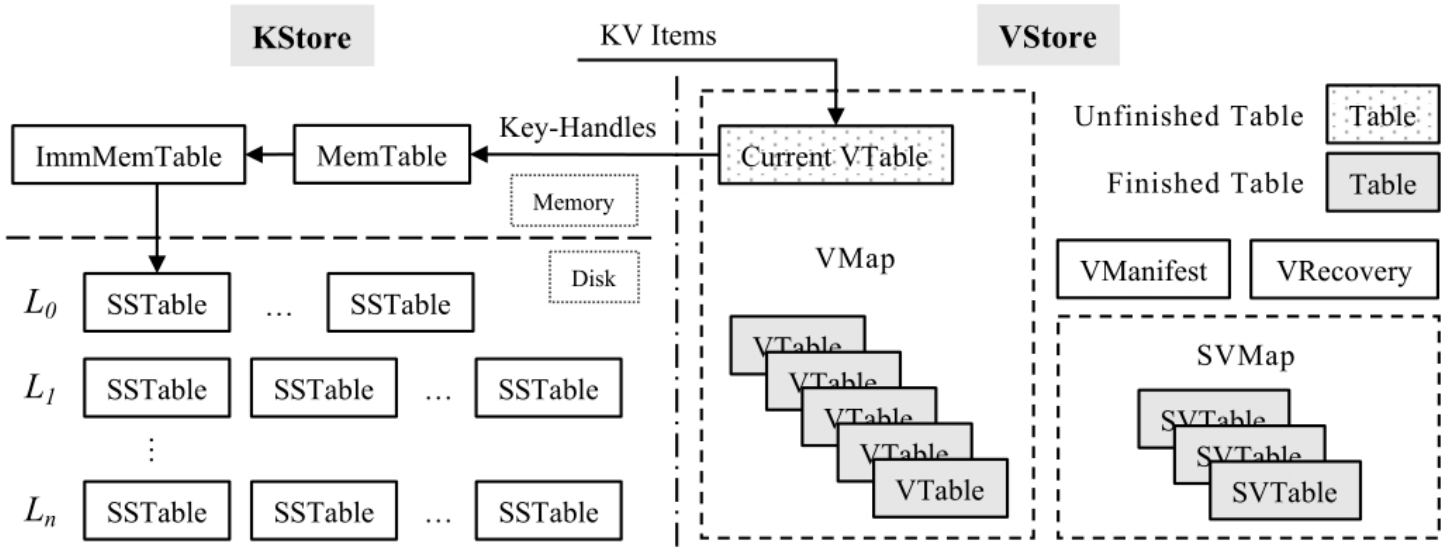
WiscKey (FAST'16)





Architecture

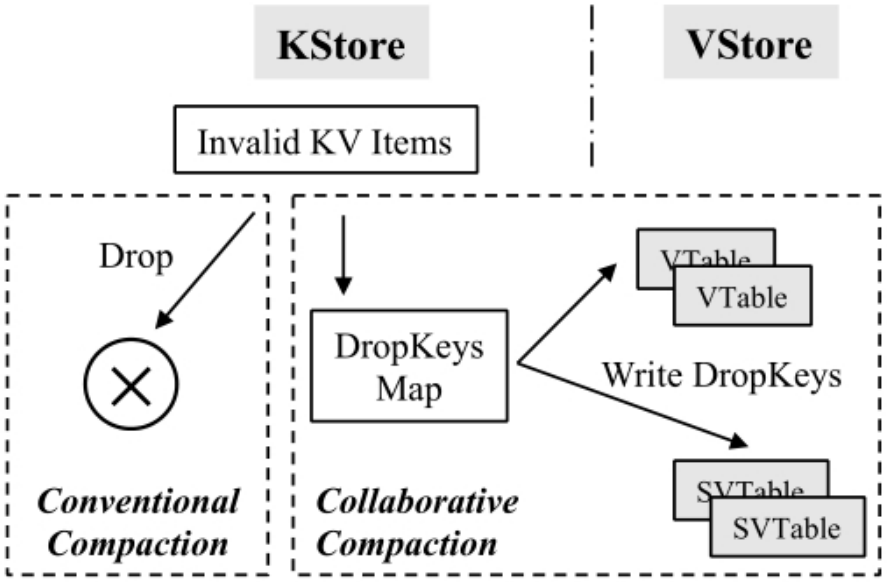
➢ **NovKV**



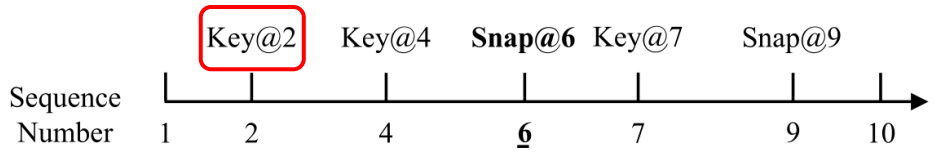
- KStore
  - LSM-Tree
  - Key-Handles
  - Compaction
- VStore
  - VTable
  - SVTable
  - Garbage Collection

Design

➢ KStore: Collaborative Compaction

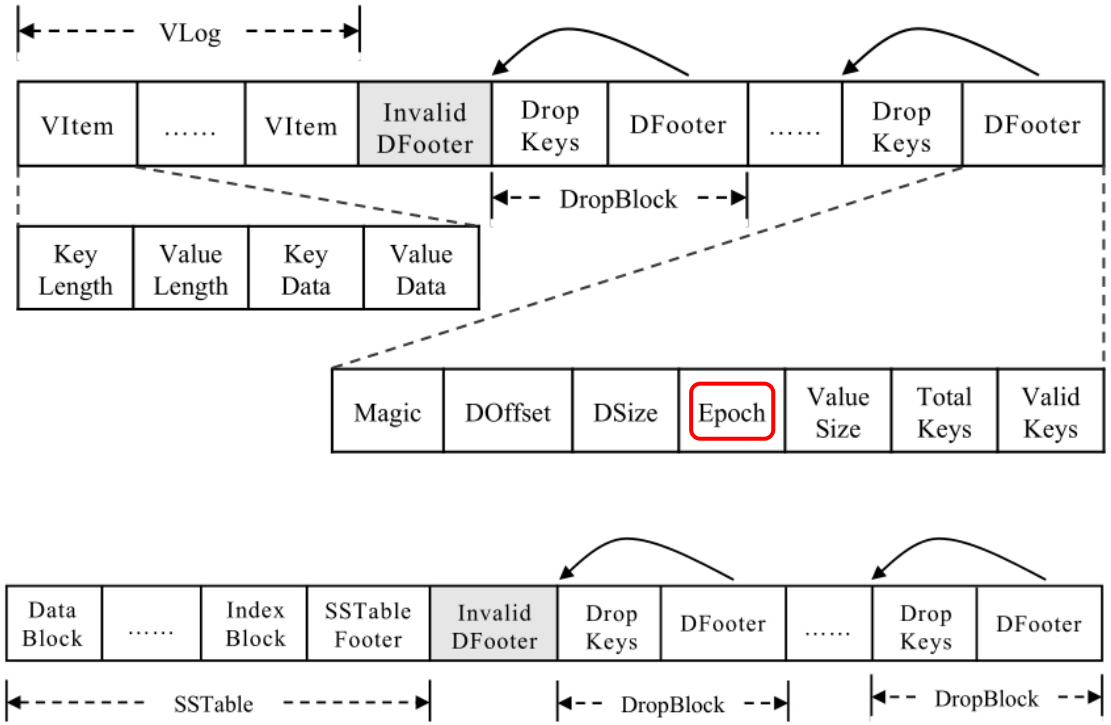


- ➢KStore compaction & VStore garbage collection
- ➢Invalid KV Items



Design

➢ VStore: Efficient Garbage Collection *



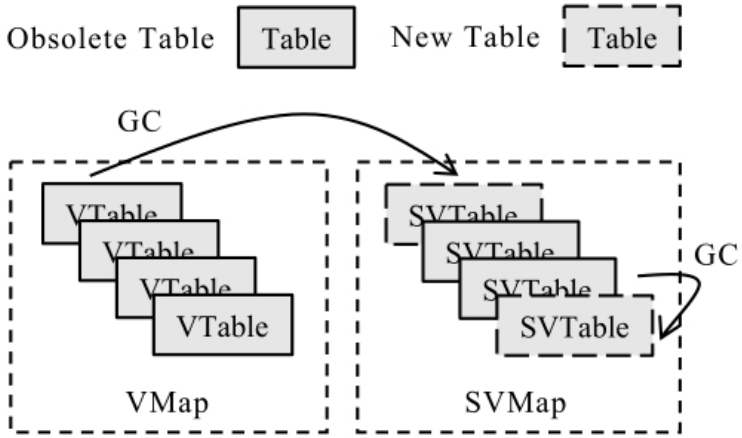
➢VTable

- VLog & unordered
- DropKeys-Map
- Epoch

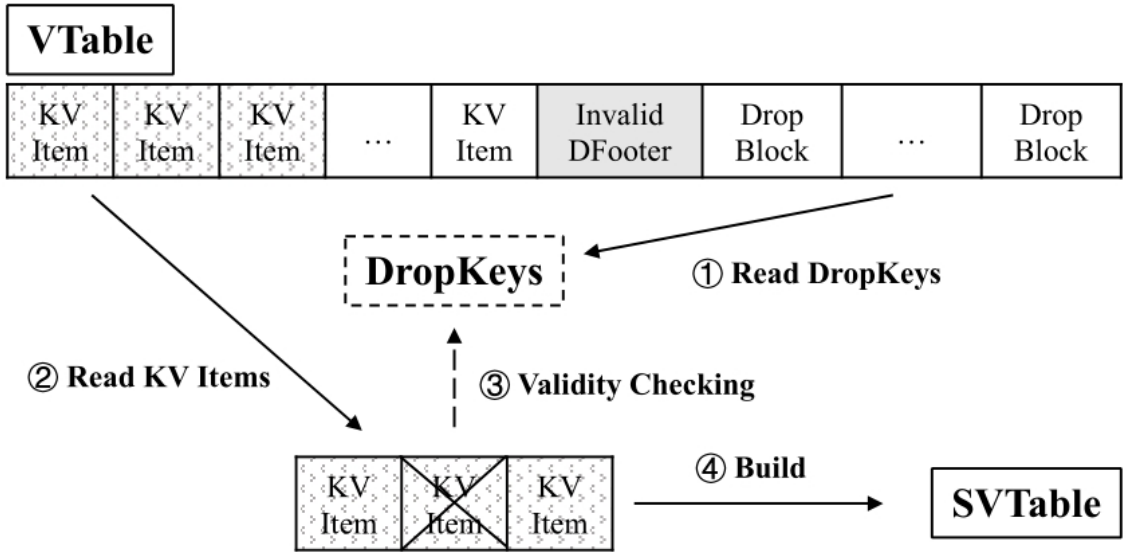
➢SVTable

- SSTable & ordered

Design



➢ VStore: Efficient Garbage Collection *



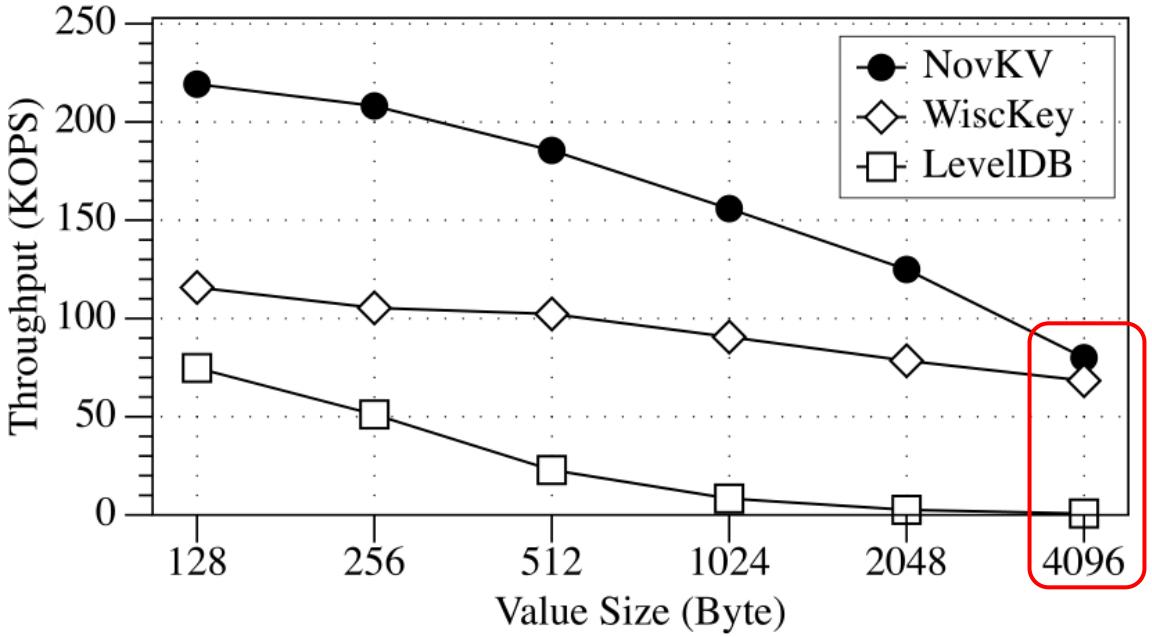
- ➢ Efficient Validity Checking
  - no need of querying KStore
- ➢ Bulid SVTable & Epoch
- ➢ New DropKeys while GC
  - Copy to SSTable

Design

- **Selective Handle Updating**
  - While NovKV searches for a value, update the KStore (Key-Value Handle) in MemTable
  - Update on request & Cost of completely memory write is low
  - Low Handle Update->Low KStore Compaction->Low VStore GC

Evaluation

➢ **Random Write**



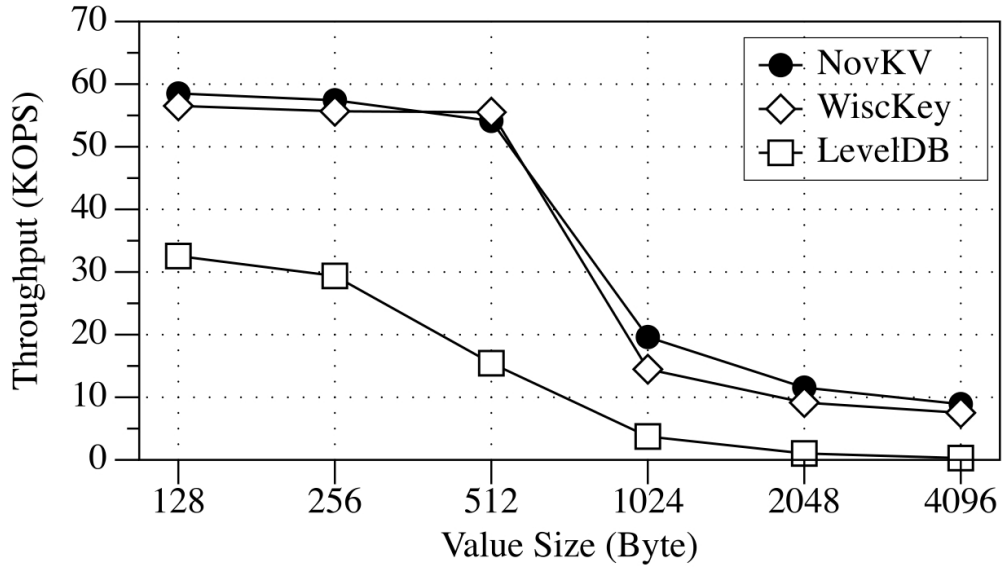
VTable and SVTable: 16MiB
GC threshold: 0.7
Insert 100M KV items
Key Size: 16B

- The throughput of NovKV is at most roughly 2× over WiscKey.
- Benefits from avoiding insertions in KStore

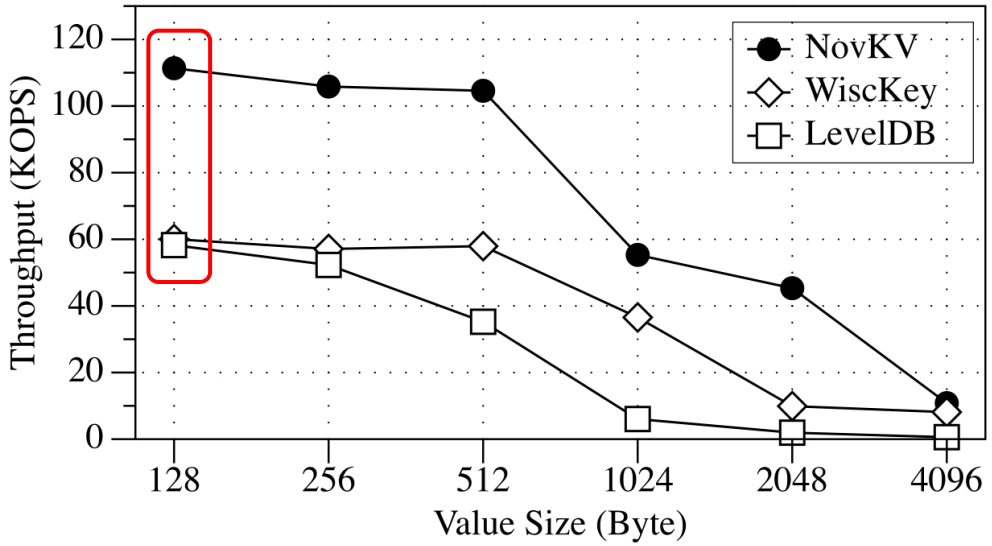
Evaluation

- **Random Read**

Random Read 10M KV items



First Read Run

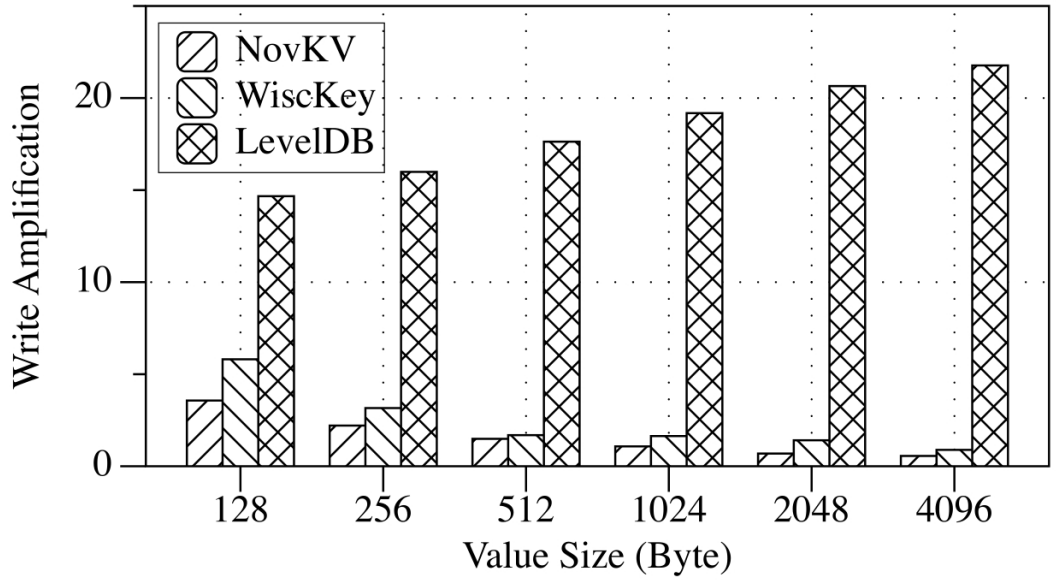


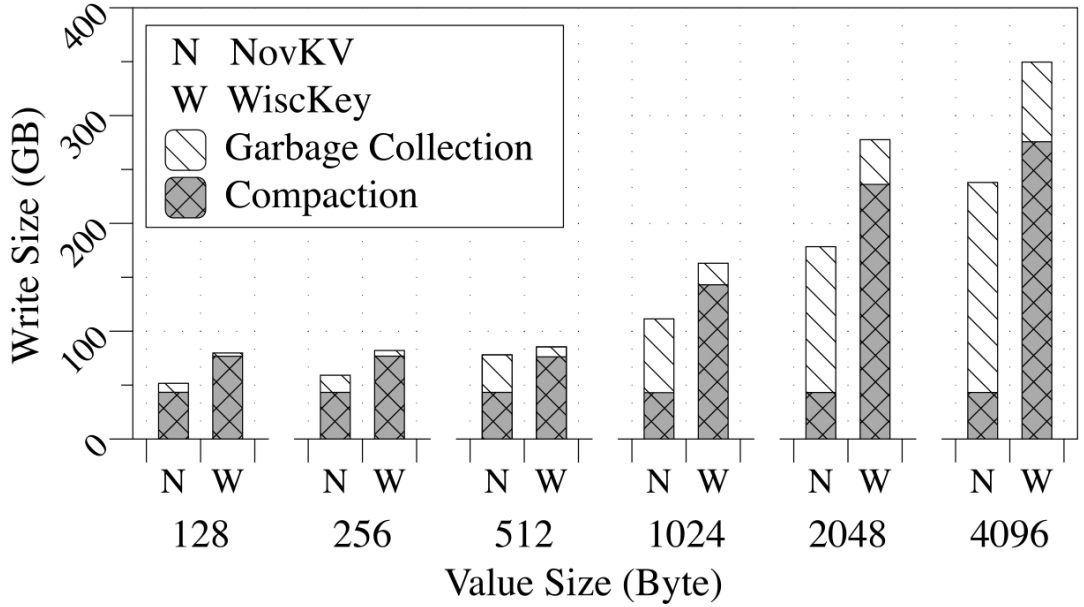
Second Read Run

- Value search & Handle Update
- Extra query overhead in VStore SSTable
- Better random read throughput of NovKV

Evaluation

➢ Write Amplification

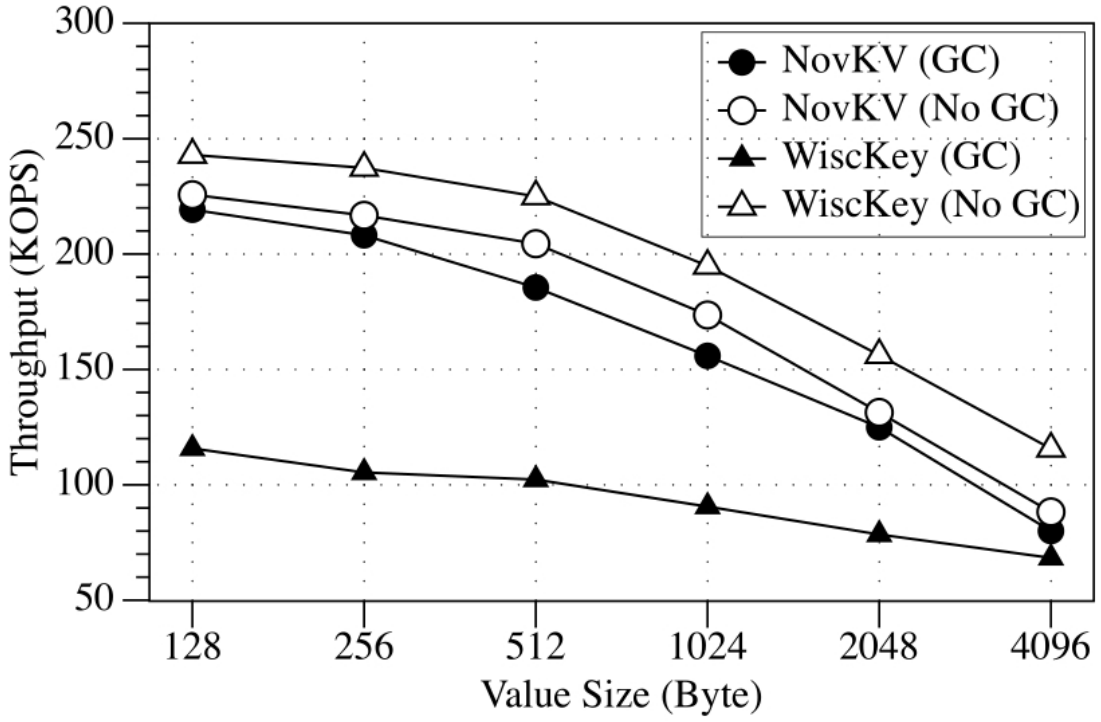




- ➢ Total write amounts / Raw data size
- ➢ Compaction & GC Write Size
- ➢ KStore Compaction & VStore GC have positive feedback interactions.

Evaluation

- **The Impact of GC**



- GC of WiscKey has very serious negative impacts on the performance.
- Benefits from the elimination of validity checking and handle updating
- GC of NovKV has slight impact on the performance.

Conclusion

- **NovKV**
  - KStore: Collaborative Compaction
  - VStore: Efficient Garbage Collection
  - Selective Handle Updating*

- **Evaluation**
  - Elimination of validity checking and handle updating
  - Better throughput & Lower write amplification

Append - 一些想法

- key 和value 分离的系统中，对于value的GC和key的update，我们可以采用message形式将信息写入VStore代替每次在KStore中的validity check
- Multi-Server Deduplication 共享key，data存在一个服务器上& 结合Pragh进行Data Migration，现在好像也是键值分离？key对应的是file recipe
- 问题：有些过程其实是可以offline进行的，这部分节省是值得的吗？