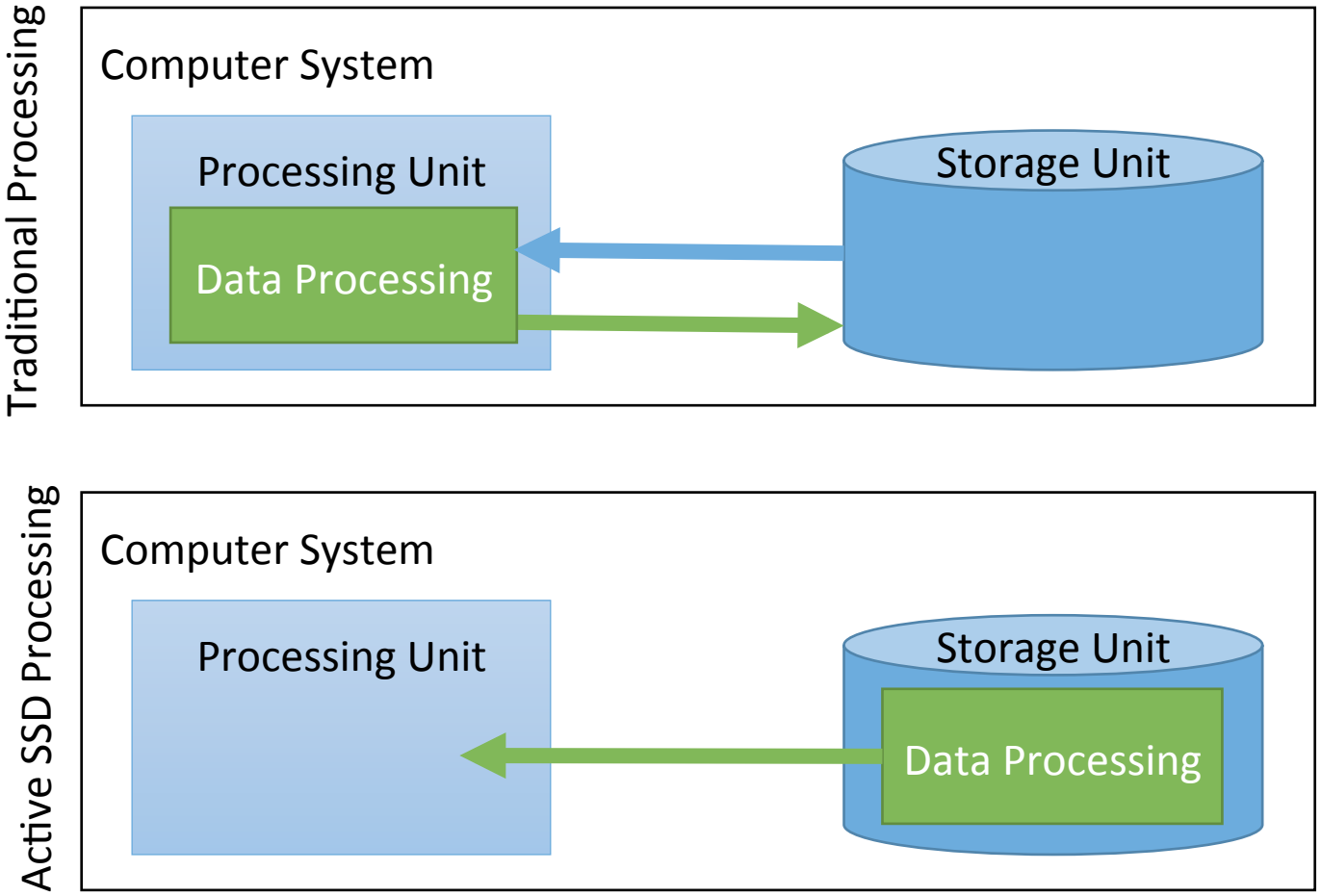
Self-Sorting SSD: Producing Sorted Data Inside Active SSDs

Luis Cavazos Quero
luis@skku.edu
Sungkyunkwan University

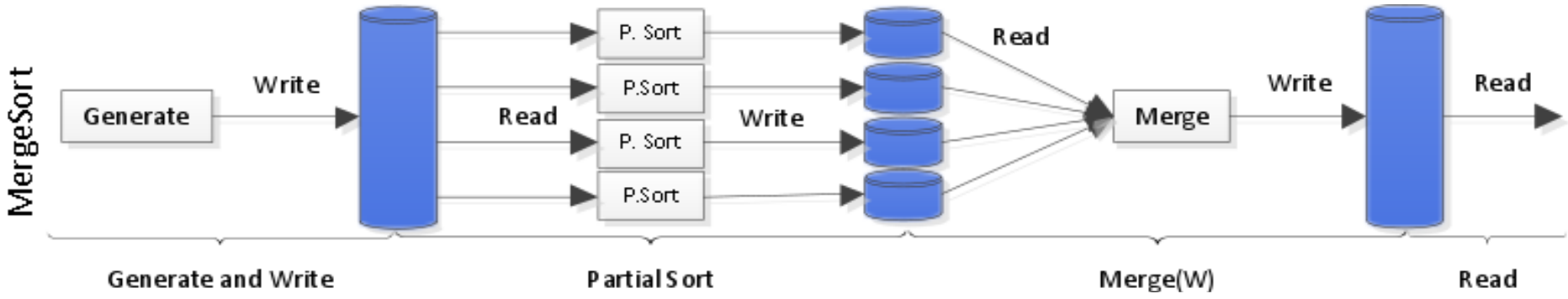
Young-Sik Lee
yslee@calab.kaist.ac.kr
KAIST

Jin-Soo Kim
jinsookim@skku.edu
Sungkyunkwan University

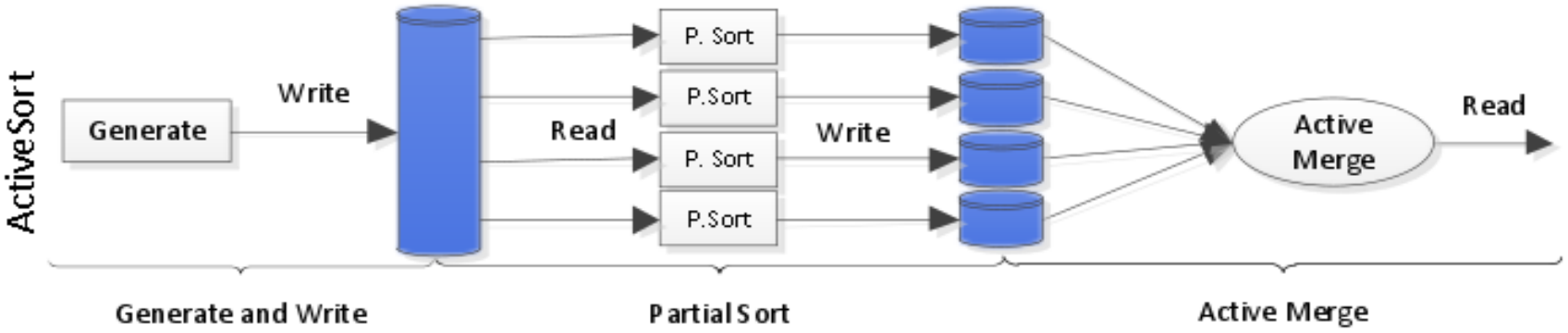
Processing Data & Active SSDs



External Sort



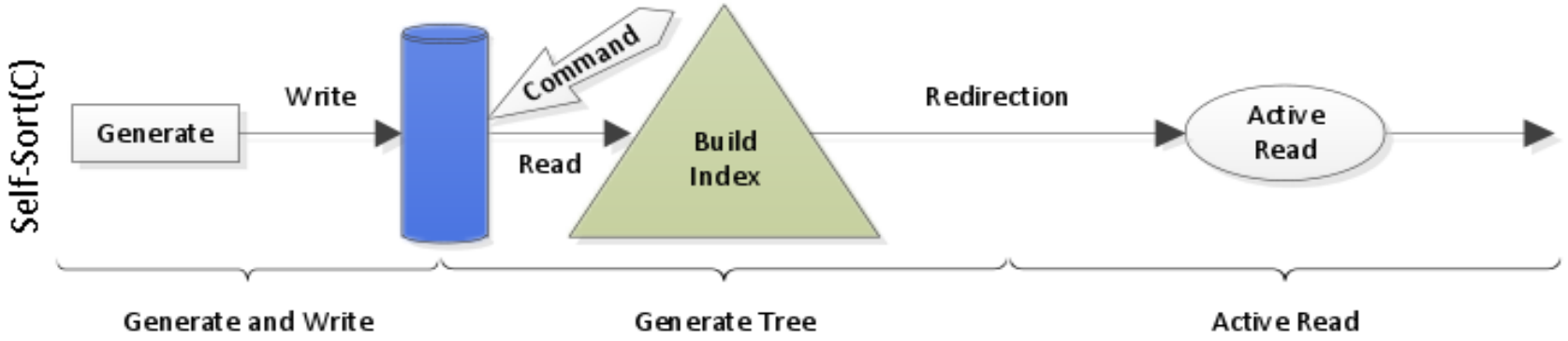
ActiveSort [HotStorage 2014*]



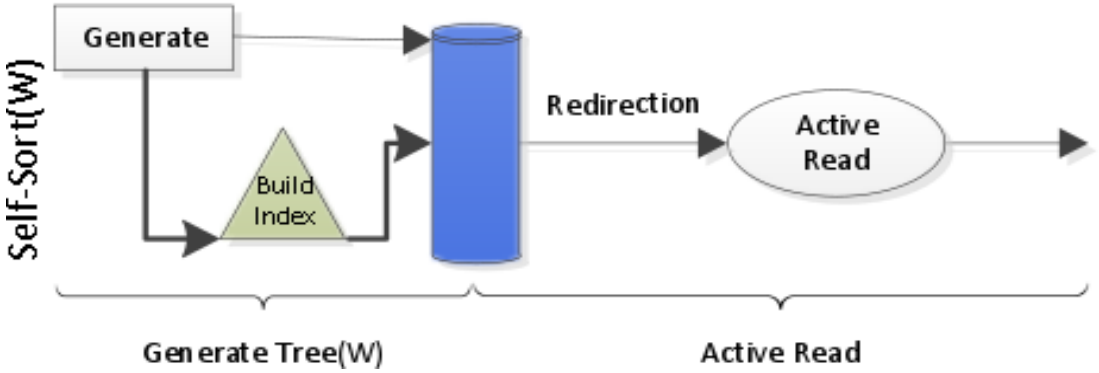
*Young-Sik Lee, Luis Cavazos Quero, Youngjae Lee, Jin-Soo Kim, and Seungryoul Maeng, "Accelerating External Sorting via On-the-fly Data Merge in Active SSDs," *Proceedings of the 6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 2014)*,'' Philadelphia, PA, USA, June 2014.

Self-Sorting SSD

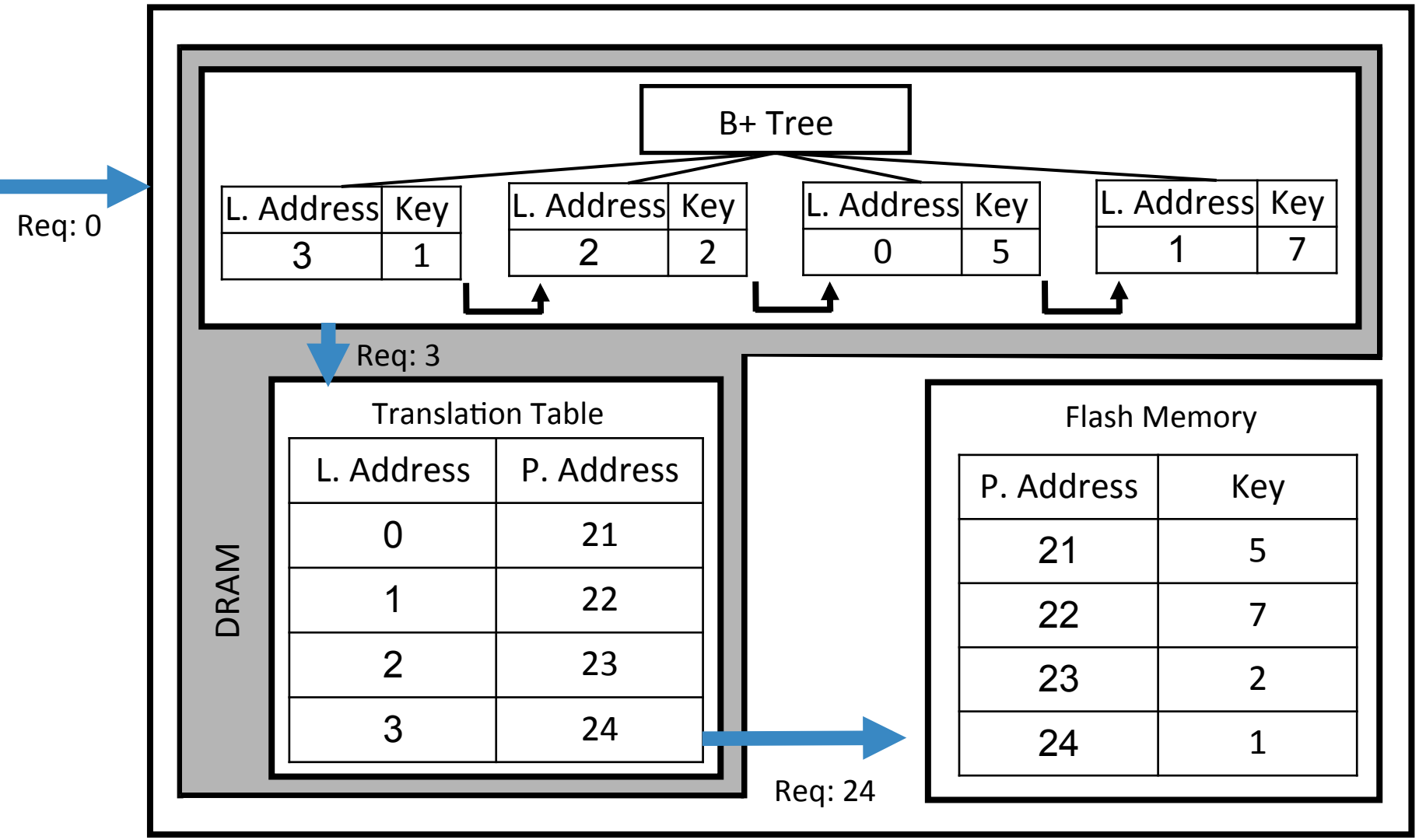
SORT-ON-COMMAND



SORT-ON-WRITE



Index – Redirection process



Self-Sorting SSD

Prototype

Jasmine OpenSSD Platform

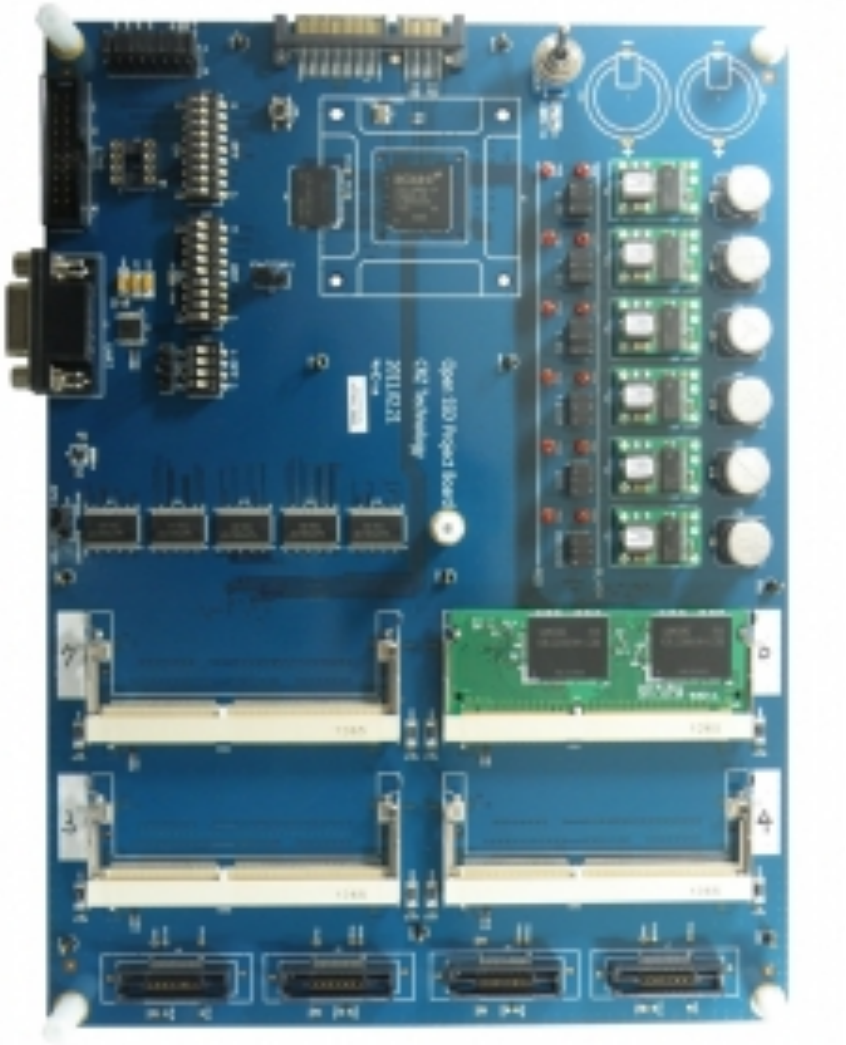
- Indilinx Barefoot SSD controller
- ARM7TDMI-S core 87.5MHz
- 96KB SRAM
- SATA 2.0 host interface (3Gbps)
- 64MB SDRAM 175MHz
- Physical page: 16 KB

Host

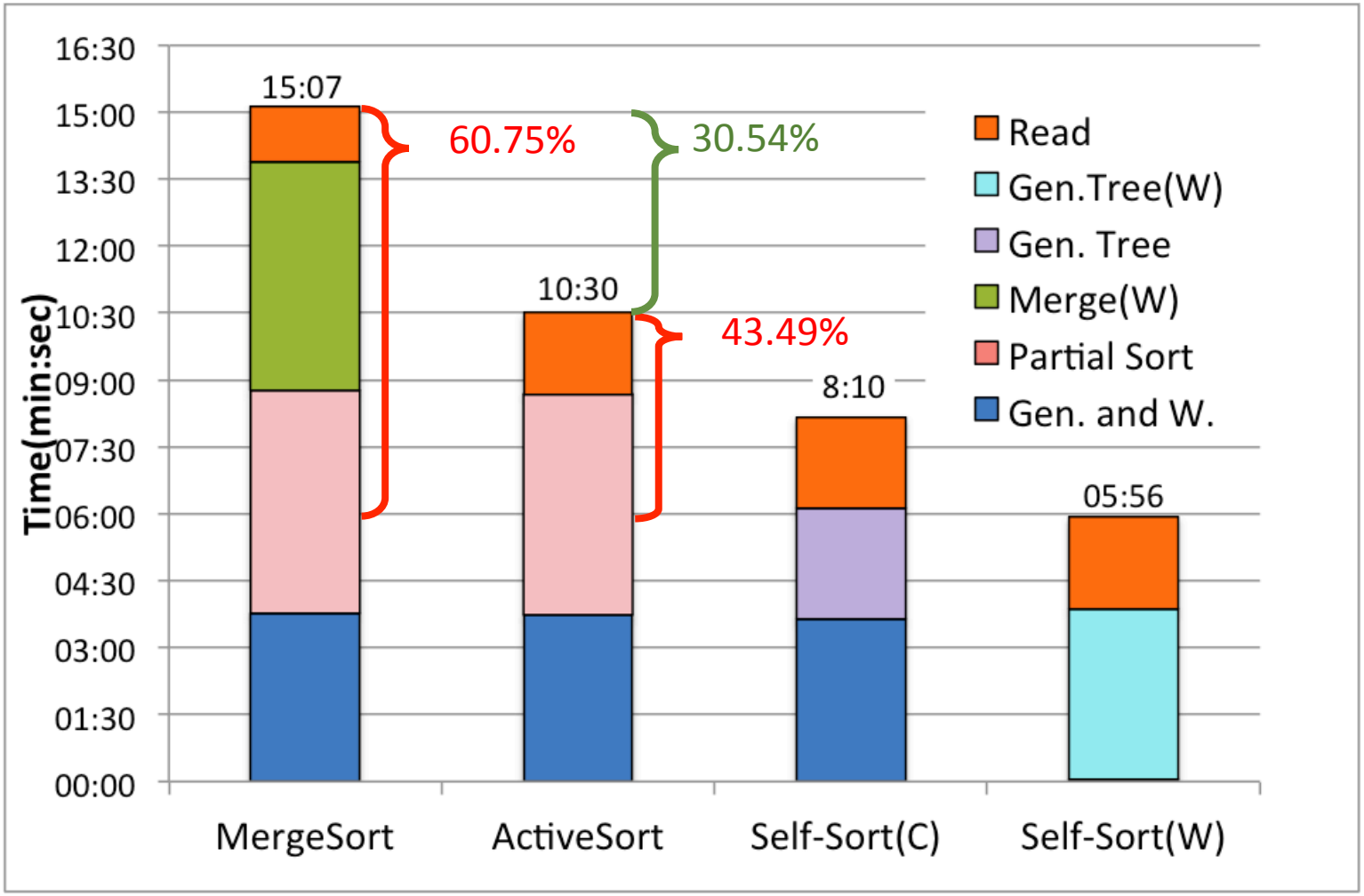
- Intel Core i5 3.4 GHz
- 16GB RAM (reduced to 3GB)
- Ubuntu 12.04
- DIRECT IO

Input data

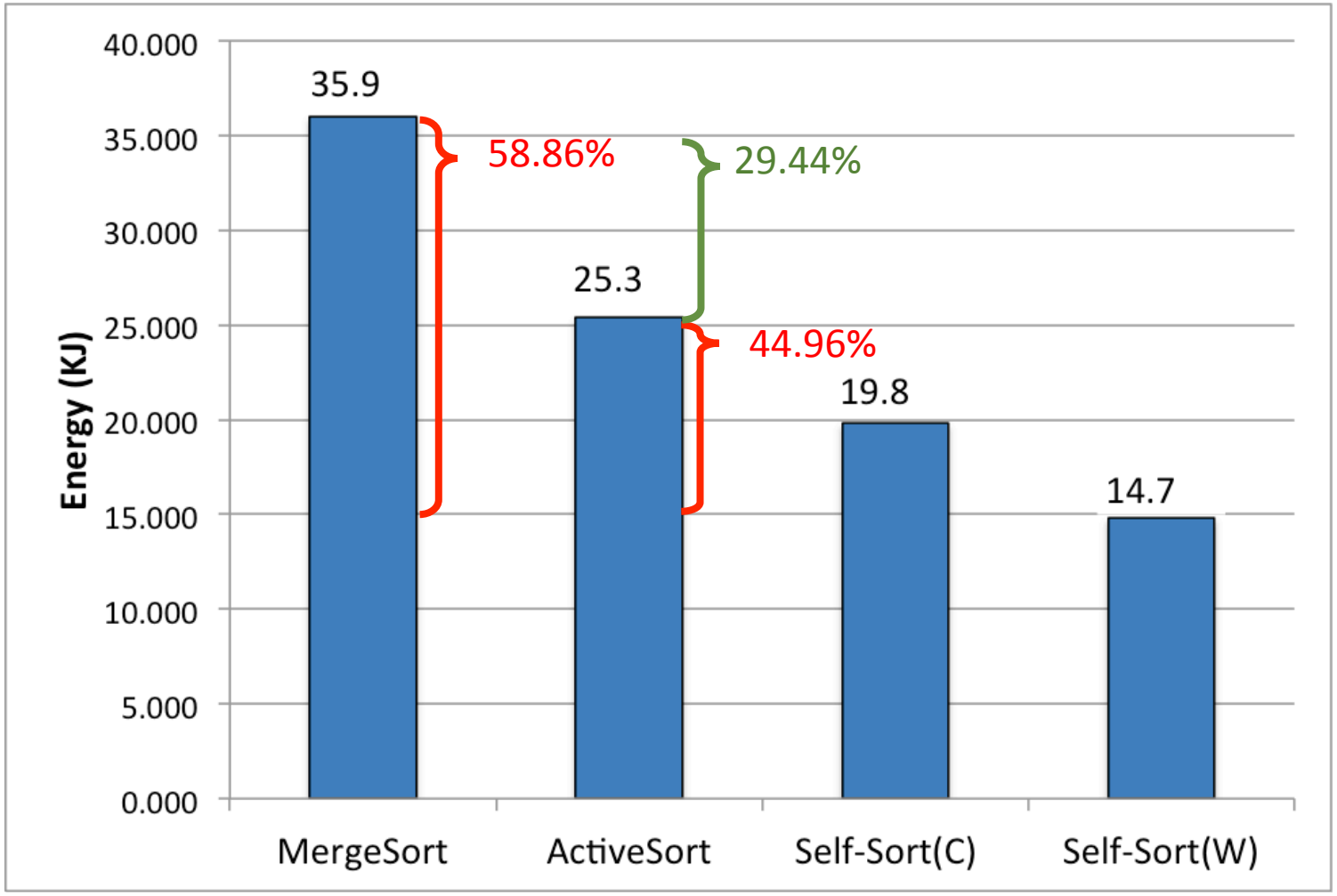
- 8GB (524,288 records)
- 16 KB record size
- 10 byte keys
- Index size 12 MB
- Fanout 128



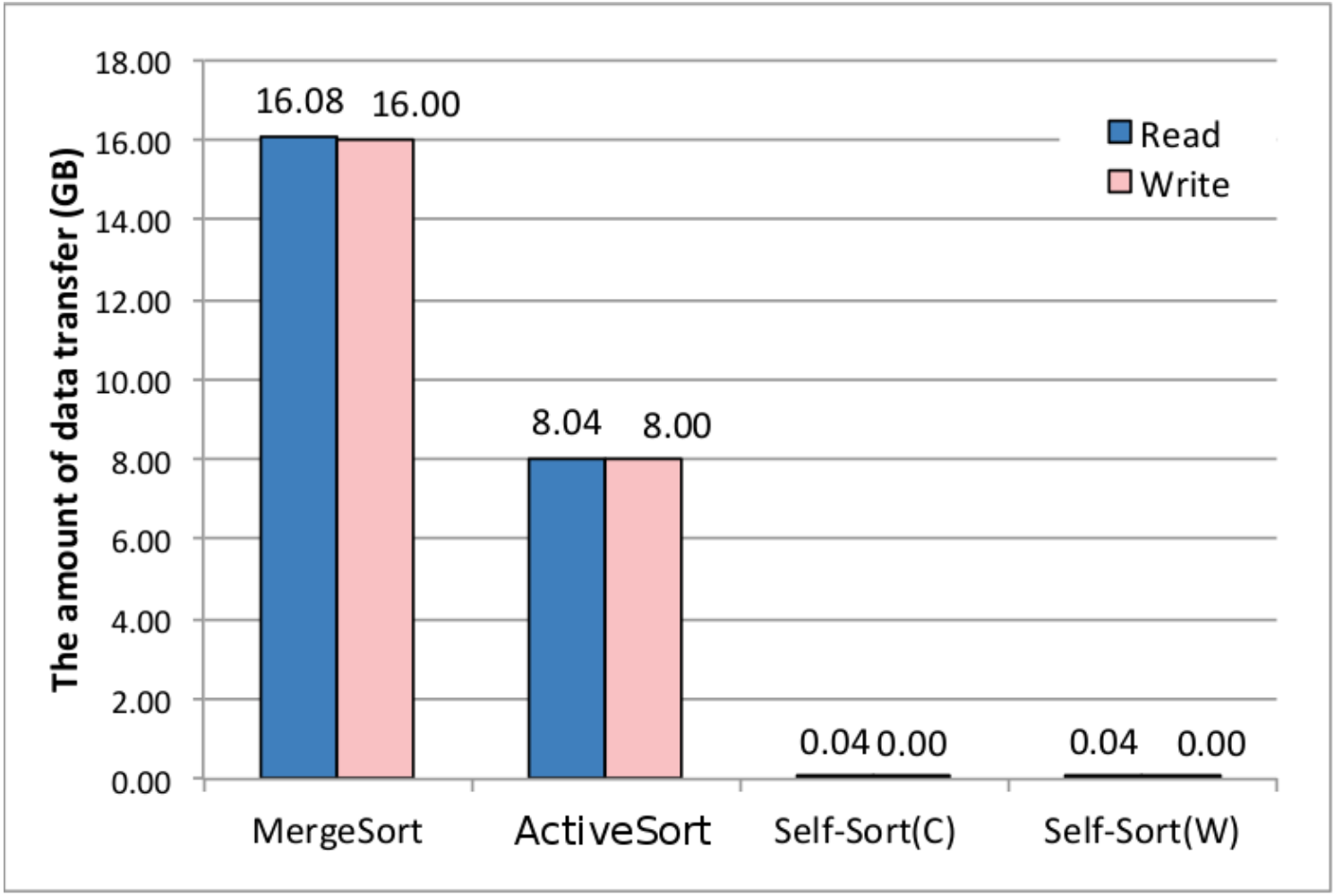
Performance



System Energy



I/O Overhead


The amount of data transfer (GB)
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
16.08
16.00
8.04
8.00
0.04
0.00
0.04
0.00
Read
Write
MergeSort
ActiveSort
Self-Sort(C)
Self-Sort(W)

Conclusions

- Completely remove write from external sort process
- Performance and SSD's lifetime improved
- More powerful SSDs will speedup the offline process and enable offloading more complex functions

Thank you!

Q/A

Active SSD – Host interface

- Standard write operation to predefined LBA
- One physical page 16 KB

Example:
Configuration/Generate/Enable command
Application

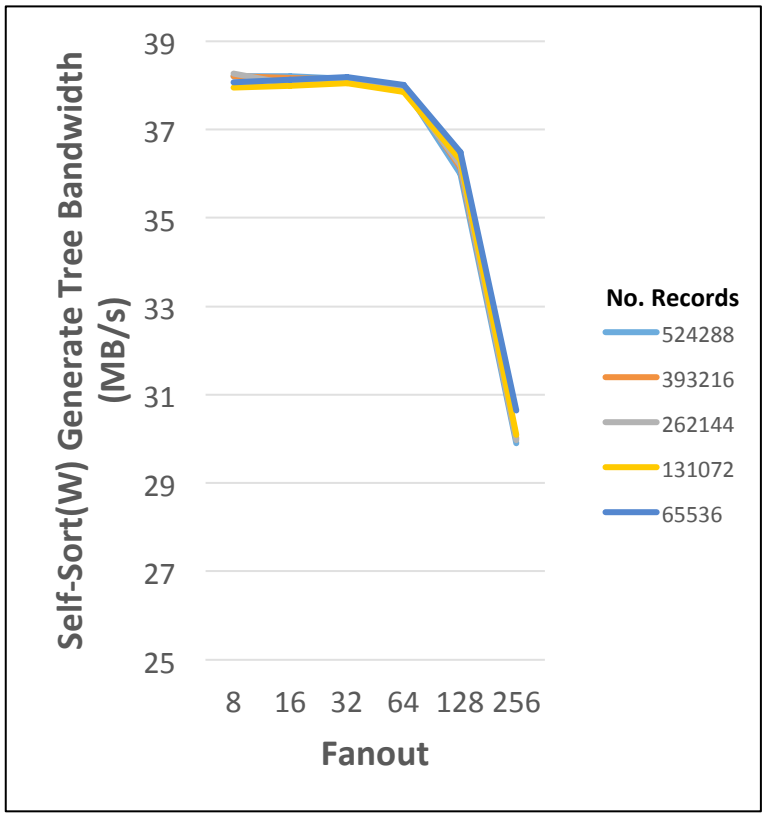
1. Allocate 16KB of memory
2. Set values

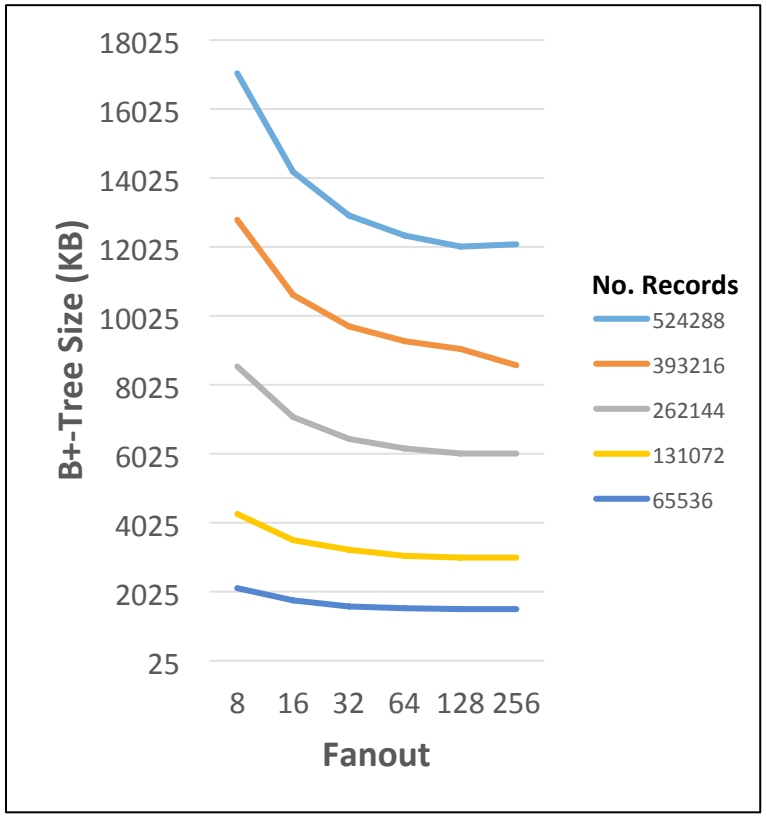
[<record size><key size><key offset><fanout><enable SOW><enable redirection><etc.>]

3. Write to Active SSD

Fanout

- No. of records inserted has no effect in insertion bandwidth
- As fanout increases performance drops
- As fanout increases B+-tree size is more compact





Rercord Size

- Records shorter than SSD read unit decrease performance
- Problem: Data is read and discarded wastefully
- Possible solutions
  - Use flash with short read unit close to record size
  - Take advantage of locality

B+-Tree

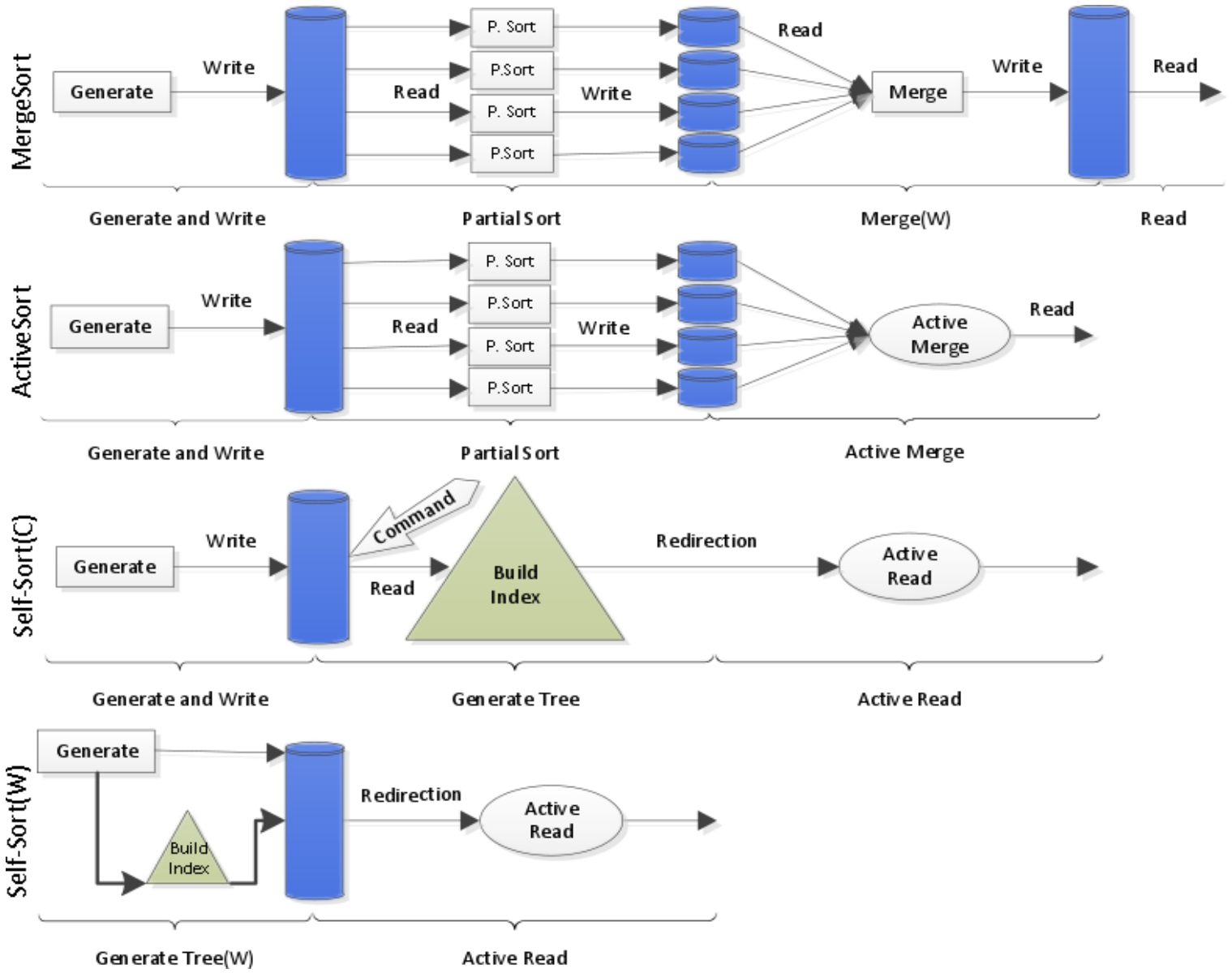
- Convenient linked list at the bottom
- Mem-copy operations are not always negligible.
- One node contains 128 keys. Serve 128 records in 1 fetch.
- If Tree was flushed into flash (big datasets) this bec omes even more relevant.

Host Index Sort

- Can also avoid write operations

- Still, Self-Sorting SSD can
  - Free host resources (CPU, Mem, IO)
  - Scalability
  - Reduce number of IO requests

Evaluation


MergeSort
Generate
Write
Read
P. Sort
P.Sort
P. Sort
P.Sort
Write
Read
Merge
Write
Read
Generate and Write
Partial Sort
Merge(W)
Read
ActiveSort
Generate
Write
Read
P. Sort
P.Sort
P. Sort
P.Sort
Write
Active Merge
Read
Generate and Write
Partial Sort
Active Merge
Self-Sort(C)
Generate
Write
Command
Read
Build Index
Redirection
Active Read
Generate and Write
Generate Tree
Active Read
Self-Sort(W)
Generate
Build Index
Redirection
Active Read
Generate Tree(W)
Active Read