DB101 – Course overview

Goetz Graefe – Madison, Wis.

Why use databases and database software?

Sharing structured data

1. Reliable storage – protection against data loss
2. Privacy protection, security, and retention
3. Concurrency control – protection of data consistency
4. Schema – common understanding of the bits and bytes
5. Physical data independence – changing storage formats, automatic mapping from tables to indexes, files, etc.
6. Controlled redundancy – consistency
7. Query processing – simple data transformations from storage to application

What touches database research & development?

- Programming languages, parsing, data abstraction
- Planning (under uncertainty), statistics, sampling
- Algorithms & data structures
- Storage, storage structures, file systems
- Operating systems, scheduling, resource management
- Networking, distributed systems
- Parallelism, high-performance computing
- Fault-tolerant computing, redundancy, failures & recovery
- Security, privacy, randomization
- Theory (normal forms, constraints, concurrency control)

Course agenda

1. Sorting
2. Transactions
3. Distributed commit
4. Storage formats
5. Consistency checking
6. Query processing
7. Robust query performance
8. Streaming

Topics omitted

1. Application design, deployment, testing, debugging, tuning, maintenance, regression testing
2. NoSQL databases (i.e., “not” SQL & “not only” SQL)
3. Business intelligence, OLAP, cubes, analytics
4. Security, privacy, compliance
5. Testing & deployment & monitoring at scale
6. Cloud deployments, virtual storage and processing
7. Self-management & auto-tuning, automatic indexes & constraints
8. Performance metrics, benchmarks, regression testing
9. Data cleaning, entity matching, etc.
10. Machine learning: ML for DB, DB for ML
11. Database theory, database design, serializability
12. Database machines, hardware support
13. Disaster preparedness & recovery & testing

CS 764 entry quiz

1. How do you spell SQL?
2. What is the most central concept in relational databases?
3. What is an integrity constraint?
4. What is a normal form?
5. What is a b-tree?
6. What is physical data independence?
7. What is physical database design?
8. What is a join of two tables?
9. What algorithms can compute a join?

CS764 fall 2023: implement an external merge sort

- 1 M × 50 B = 50 MB, 2.5 M × 50 B = 125 MB
  12 M × 1 KB = 12 GB, 120 M × 1 KB = 120 GB
- 1 CPU core, 1 MB cache, 100 MB DRAM
  SSD: 10 GB capacity, 0.1 ms latency, 100 MB/s bandwidth
  HDD: ∞ capacity, 10 ms latency, 100 MB/s bandwidth
  Emulate SSD + HDD, report total latency & transfer time
- Extra credit: logic & performance evaluation for
  - in-stream (after-sort) ‘distinct’, ‘group by’, or ‘top’
  - in-sort ‘distinct’, ‘group by’, or ‘top’
- Provided: iterator template & logic

Techniques to consider in external merge sort

1. Quicksort?
2. Tournament trees [5]
3. Replacement selection?
4. Run size > memory size?
5. Offset-value coding [5]
6. Variable-size records??
7. Compression?
8. Prefix truncation?
9. Minimum count of row & column comparisons [5]
10. Cache-size mini runs [5]
11. Device-optimized page sizes [5]
12. Spilling memory-to-SSD [5]
13. Spilling from SSD to disk [5]
14. Graceful degradation
    a. into merging [5]
    b. beyond one merge step [5]
15. Optimized merge patterns [5]
16. Verifying
    a. sets of rows & values [5]
    b. sort order [5]

Using SSD and HDD effectively

Alternative 1:

- Write memory-sized runs
  - first to SSD
  - then to HDD
- Merge all runs
  - Large I/O on HDD
  - Possibly small fan-in

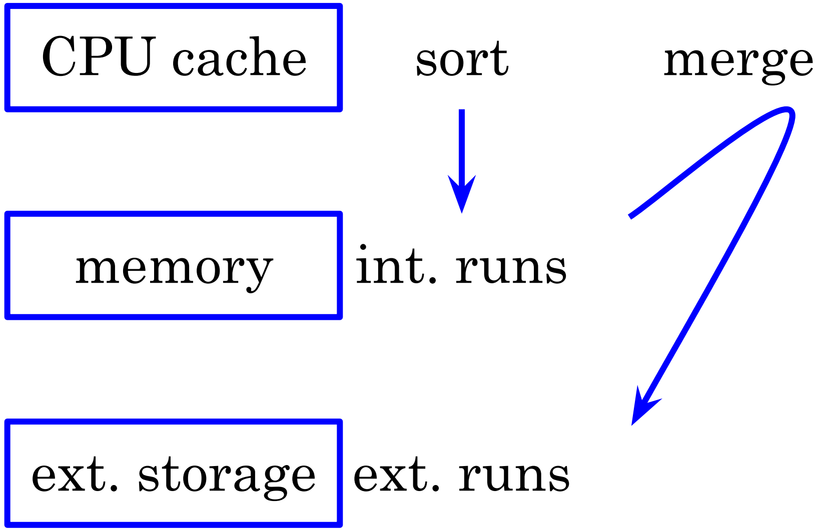
Alternative 2:

- Write memory-sized runs only to SSD
- Merge from SSD to HDD (SSD-sized runs on HDD)
- Merge from HDD via SSD
  - Large I/O HDD→SSD
  - High fan-in from SSD

Run generation in a three-level memory hierarchy

Merge cache-size internal runs
into <u>memory-size external runs</u>

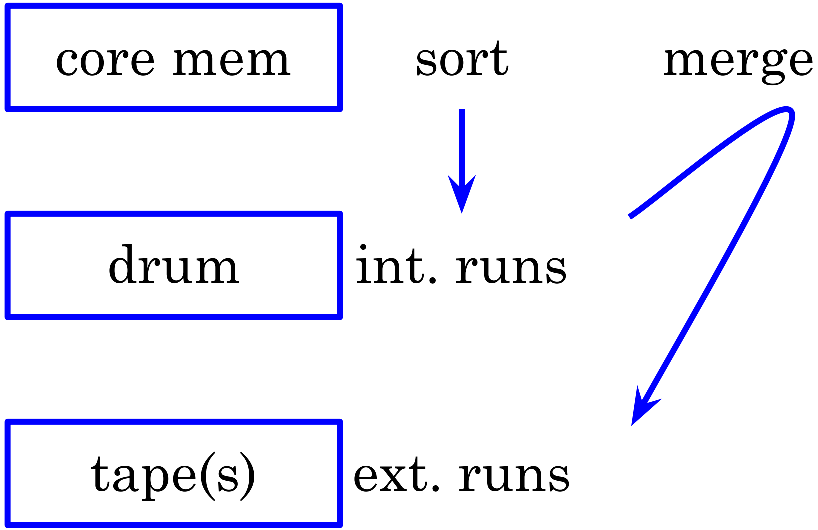
[Nyberg et al. 1996]



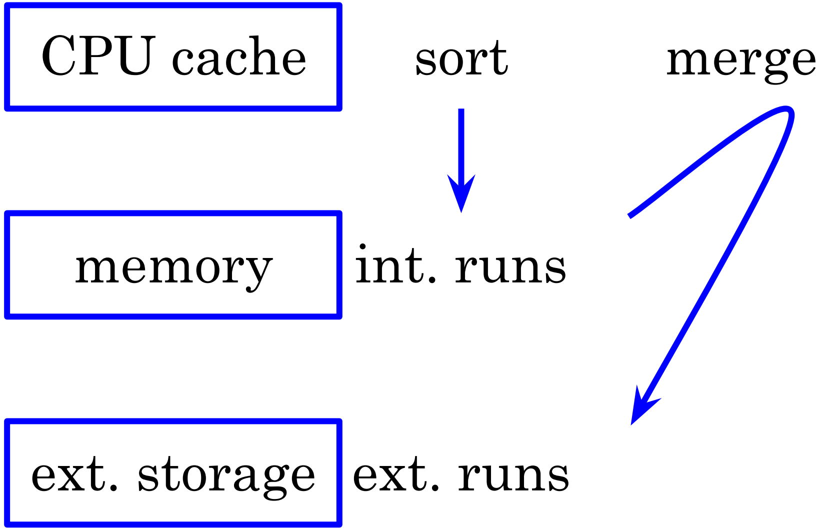
Run generation in an "ancient" memory hierarchy

Merge core-size internal runs
into drum-size runs on tape

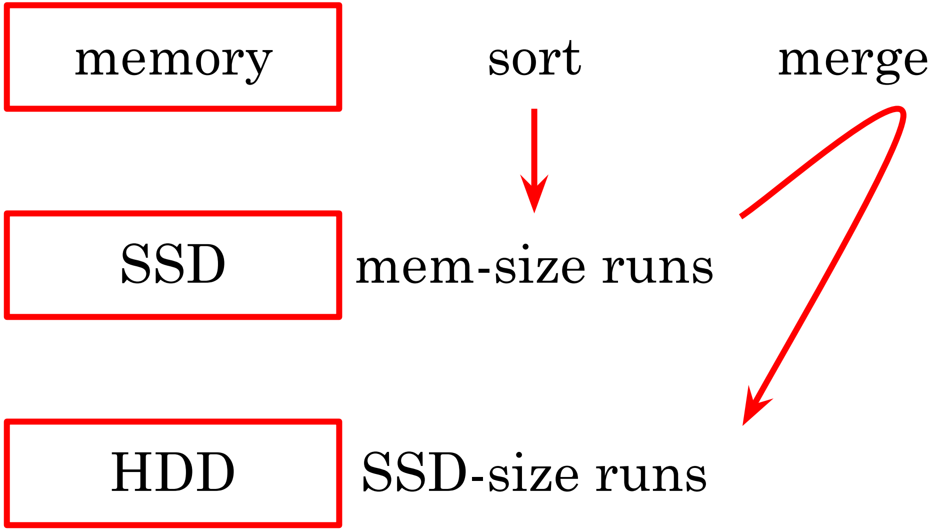
[Friend JACM 1956]



Run generation in three-level memory hierarchies



Merge memory-size runs on SSD
into <u>SSD-size runs on HDD</u>

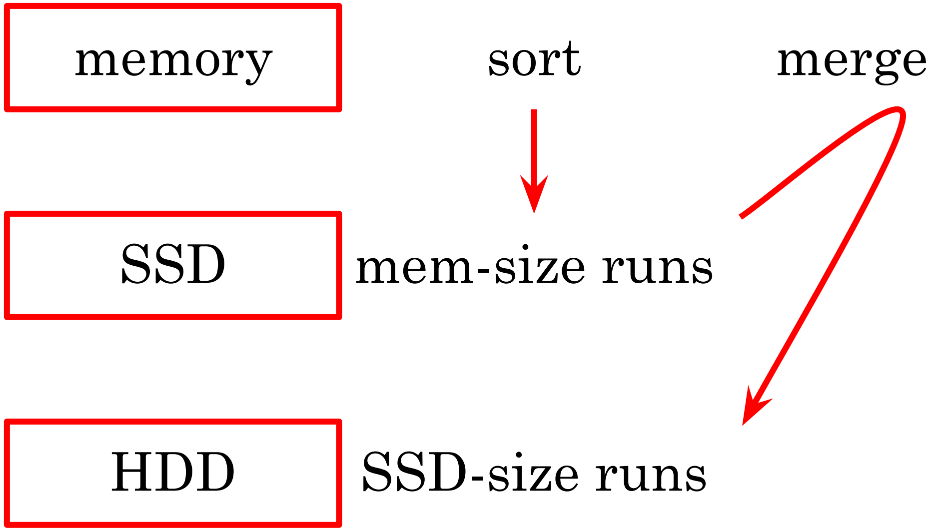


Merge cache-size internal runs
into <u>memory-size external runs</u>

Run generation in a three-level memory hierarchy

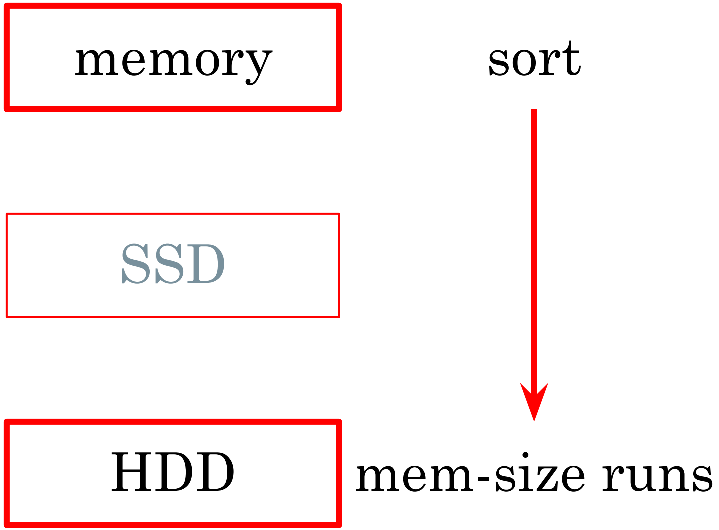
SSD continuously

- writes (from memory)
- reads (merging to HDD)



Run generation in a two-level memory hierarchy

Write memory-size runs to HDD

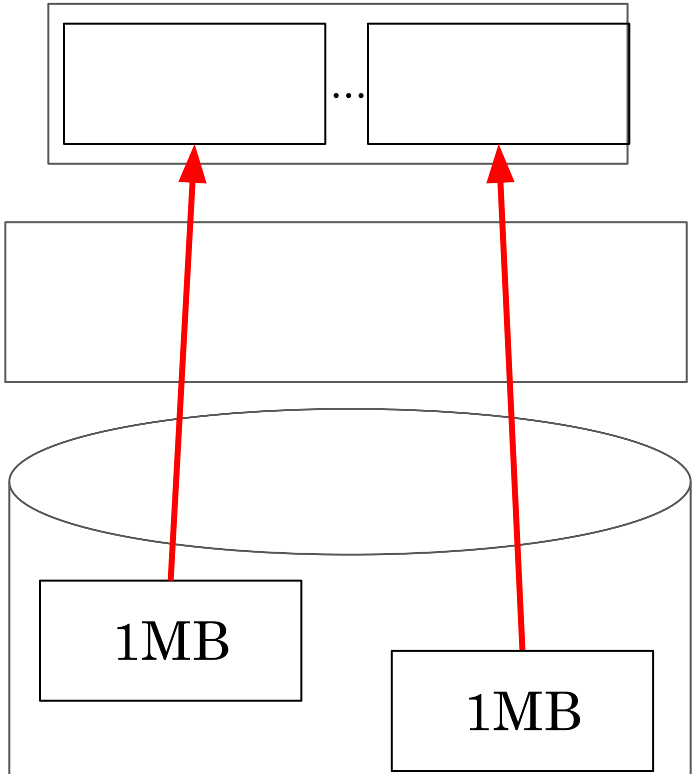


Merging in a three-level memory hierarchy (1 of 3)

100 MB memory

10 GB SSD: 8KB

"∞" HDD: 1MB/acc



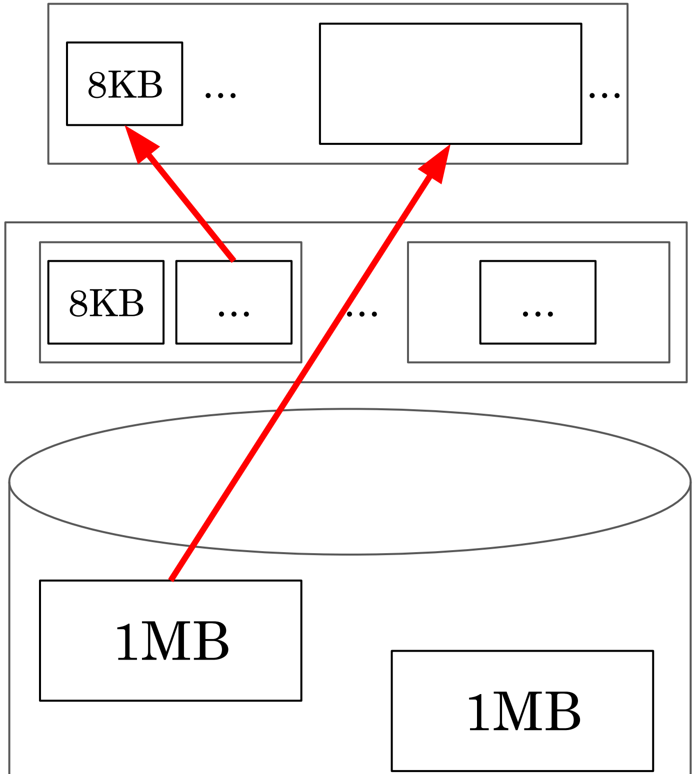
Merge fan-in F
= 100MB ÷ 1MB
= 100

Merging in a three-level memory hierarchy (2 of 3)

100 MB memory

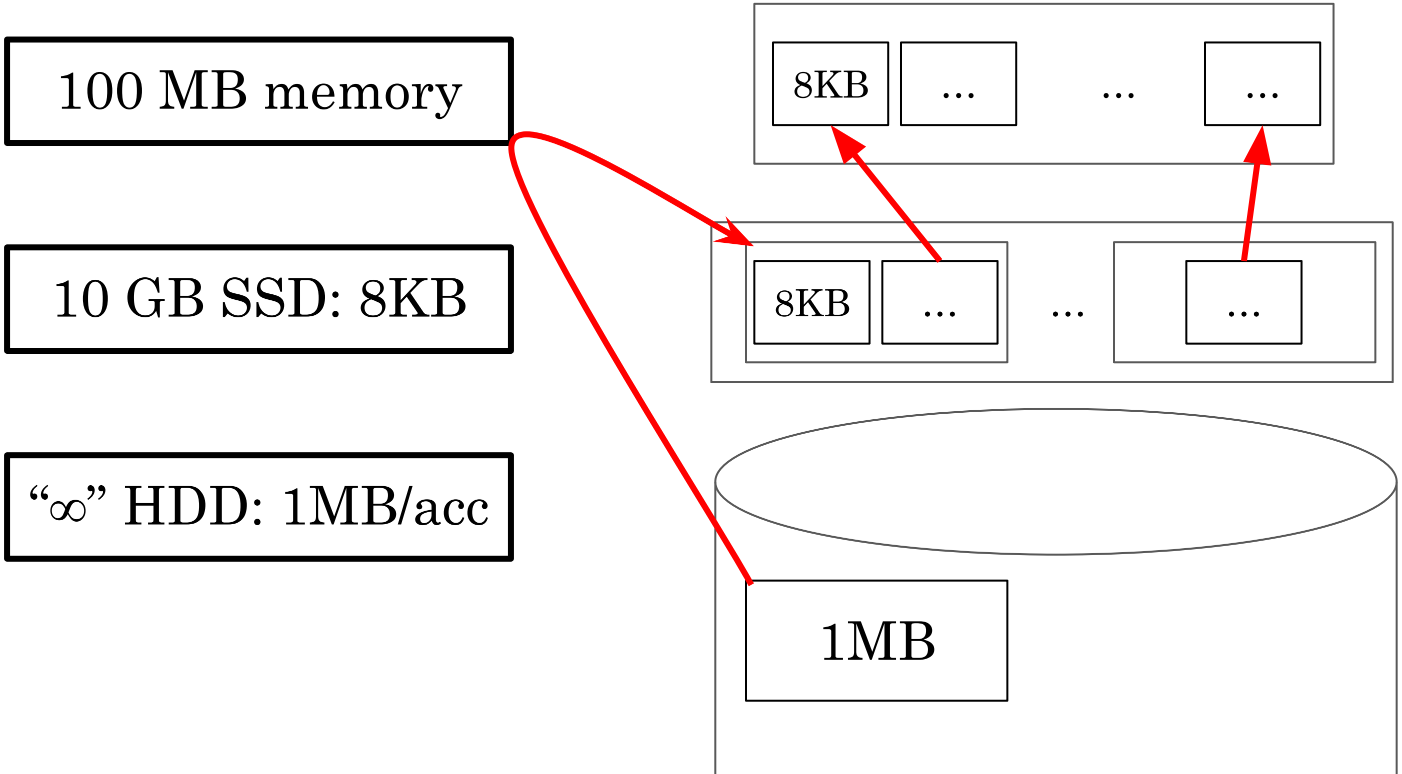
10 GB SSD: 8KB

"∞" HDD: 1MB/acc



Merge fan-in F = 10GB ÷ 100MB + (100MB – 8KB × 10GB ÷ 100MB) ÷ 1MB
= 100 + 99MB ÷ 1MB
= 199

Merging in a three-level memory hierarchy (3 of 3)



Merge fan-in F = min (100MB ÷ 8KB, 10GB ÷ 1MB) = 10K