

Data Representation & External Sorting

INF 551

Wensheng Wu

Outline

- Representing data



- How are tables stored on storage devices?

- External Sorting

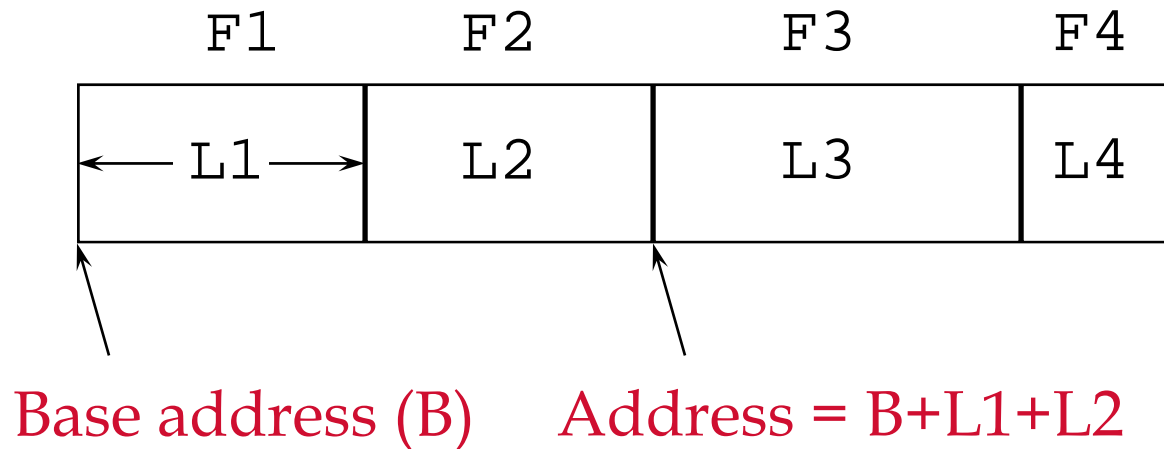
- How to sort 1TB data using 1GB of memory?

Representing Data Elements

- Relational database elements:

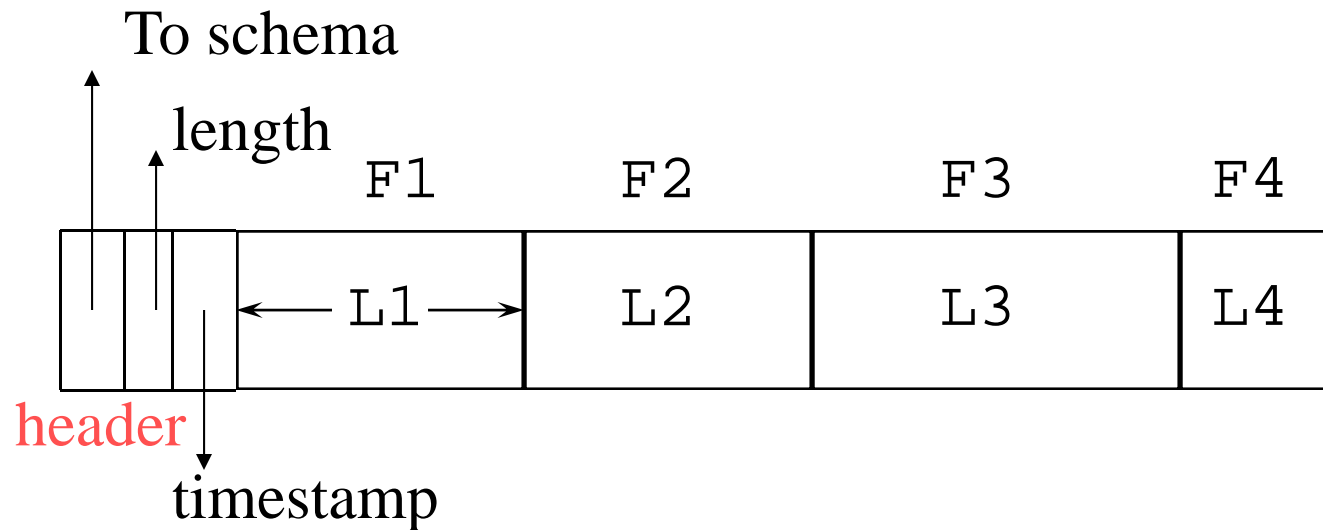
```
CREATE TABLE Product (  
  pid INT PRIMARY KEY,  
  name CHAR(20),  
  description VARCHAR(200),  
  maker CHAR(10) REFERENCES Company(name))
```
- A tuple is stored as a "record"

Record Formats: Fixed Length



- Information about field types is the same for all records in a file; stored in *system catalogs*.
- **Note the importance of schema information!**

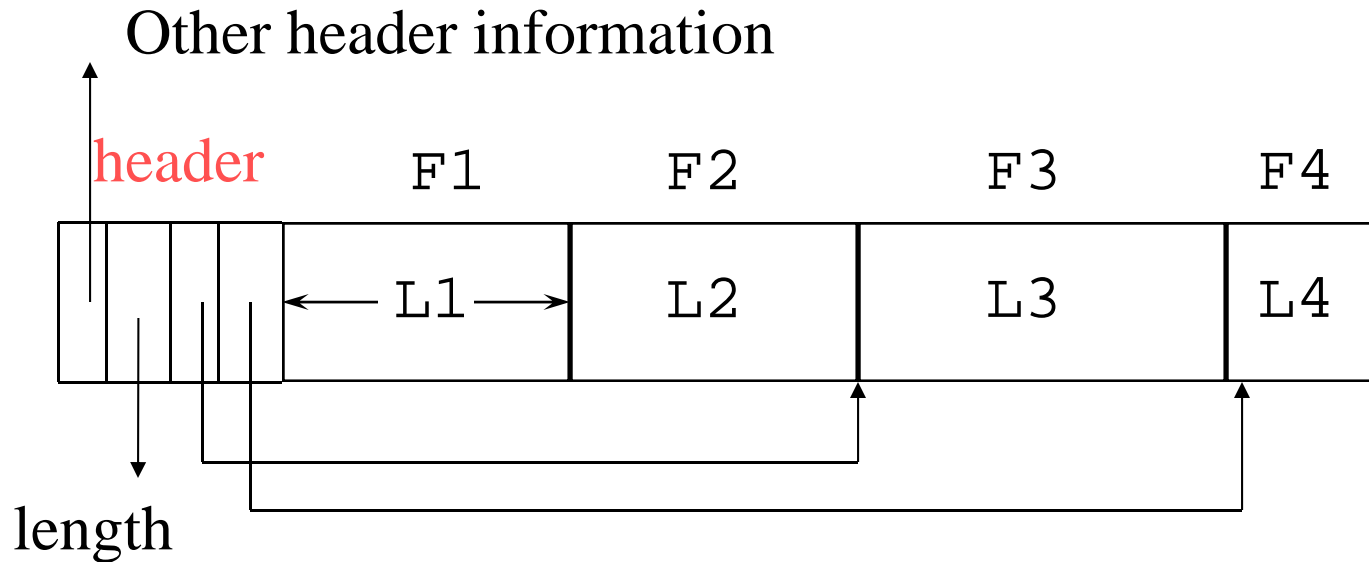
Record Header



Header:

- Pointer to schema: help finding fields
- Length: so we know where the record ends w/o consulting schema
- Timestamp: time when record last modified or read

Variable Length Records



Place the fixed fields first: F1, F2

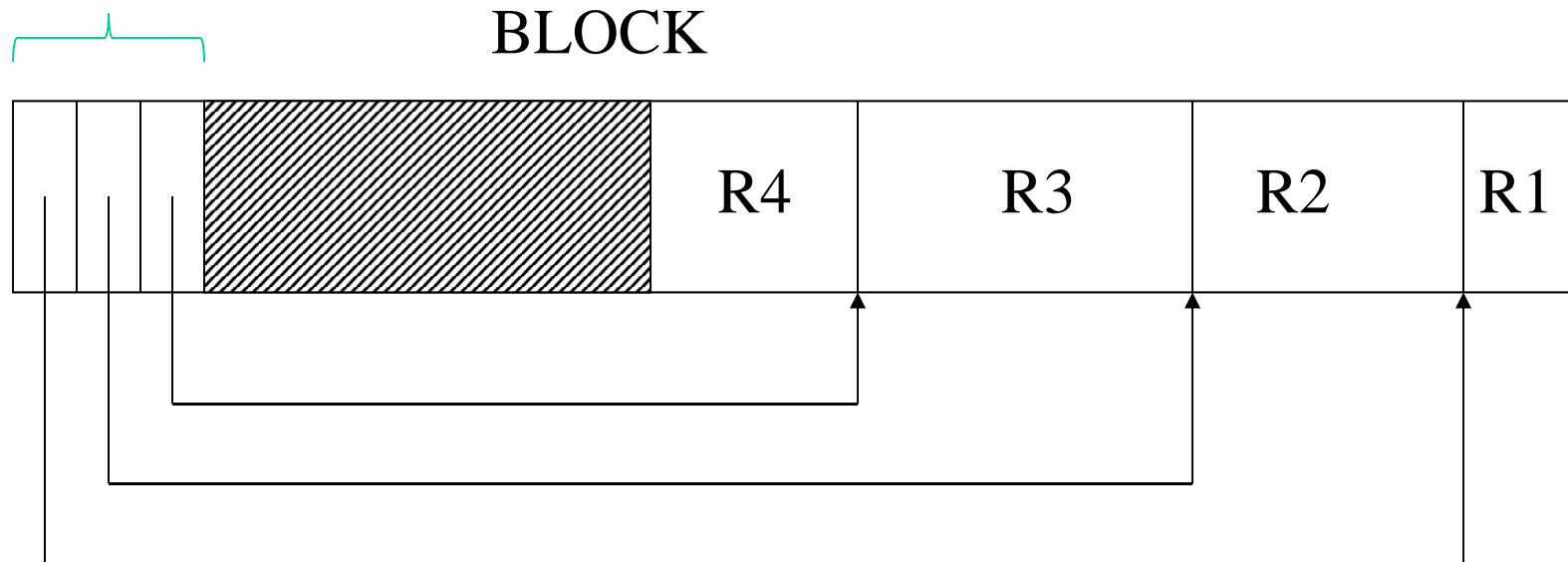
Then the variable length fields: F3, F4

Note: actually no need for pointer to F3, why?

Storing Records in Blocks

- Blocks have fixed size (typically 4KB)
 - But records may have variable-length

Offset table (slot directory)

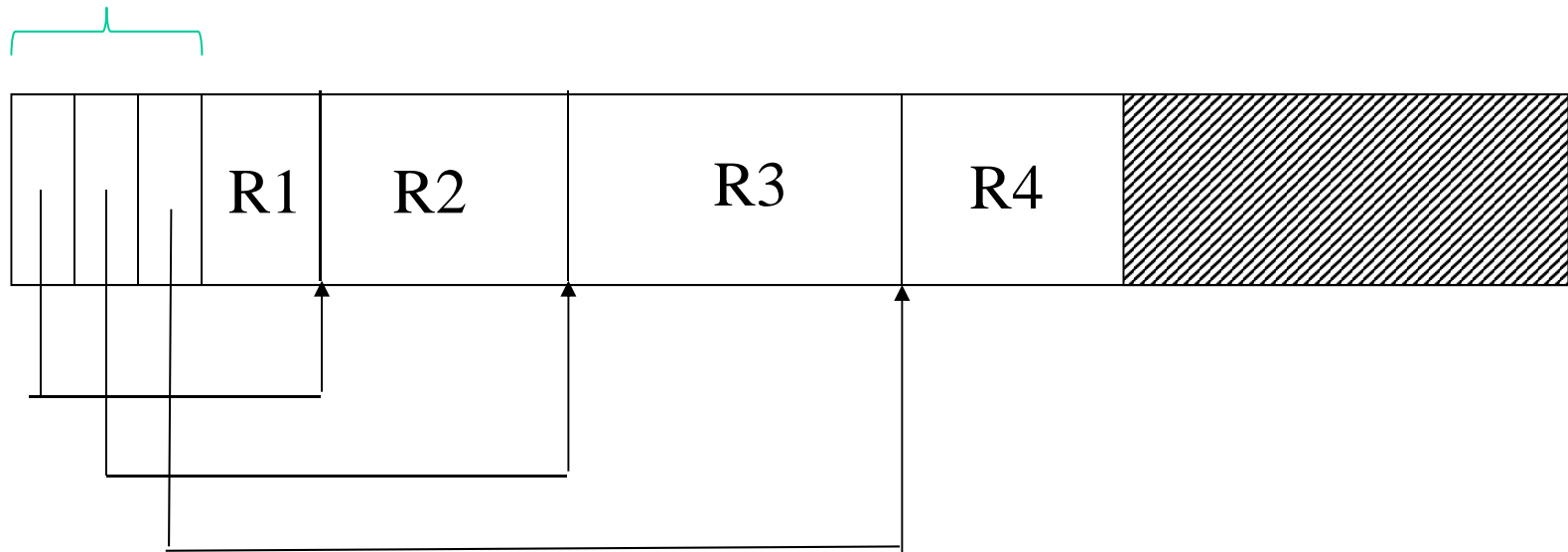


Why are records placed from the end?


Problem with this design?

- Records right after slot directory
- Free space after all records

Offset table (slot directory)



Outline

- Representing data
 - How are tables stored on storage devices?
- External Sorting 
 - How to sort 1TB data using 1GB of memory?

The I/O Model of Computation

- In main memory algorithms:
 - we care about CPU time
- In databases
 - time is dominated by I/O cost
- Assumption: cost is given only by I/O
- Consequence: need to redesign certain algorithms, e.g. sorting

Notes

- A block on storage devices loaded into a **page** in main memory
 - We sometimes interchange page with block
- Buffer pages
 - Often refer to pages in main memory used to store input, output, and intermediate data for an algorithm
- Run: a **sorted sublist** of input data

Notes

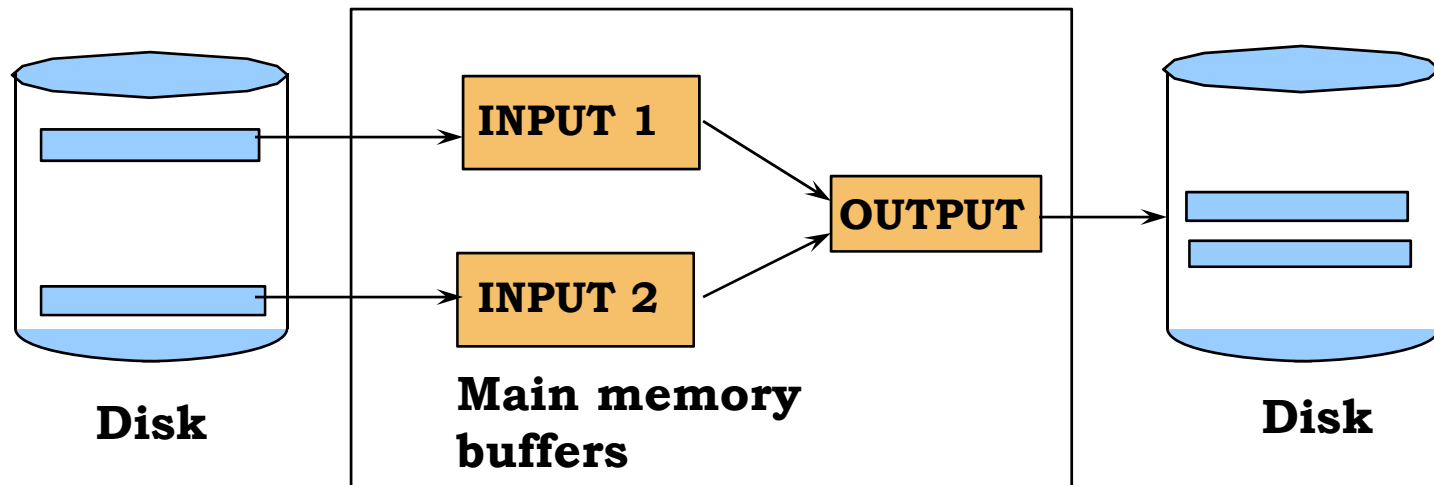
- Make a pass through data:
 - Loading the entire data from disk once

Sorting

- Illustrates the difference in algorithm design when your data is not in main memory:
 - Problem: sort 1TB of data with 1GB of RAM
- Arises in many places in database systems:
 - Data requested in sorted order (ORDER BY)
 - Needed for grouping operations
 - First step in sort-merge join algorithm
 - Duplicate removal
 - Bulk loading technique for creating B+-tree indexes

2-Way Merge-sort: Requires 3 Buffers

- Pass 0: Read a page, sort it, write it
 - only one buffer page is used
- Pass 1, 2, ..., etc.: merging two runs at a time
 - three buffer pages used.

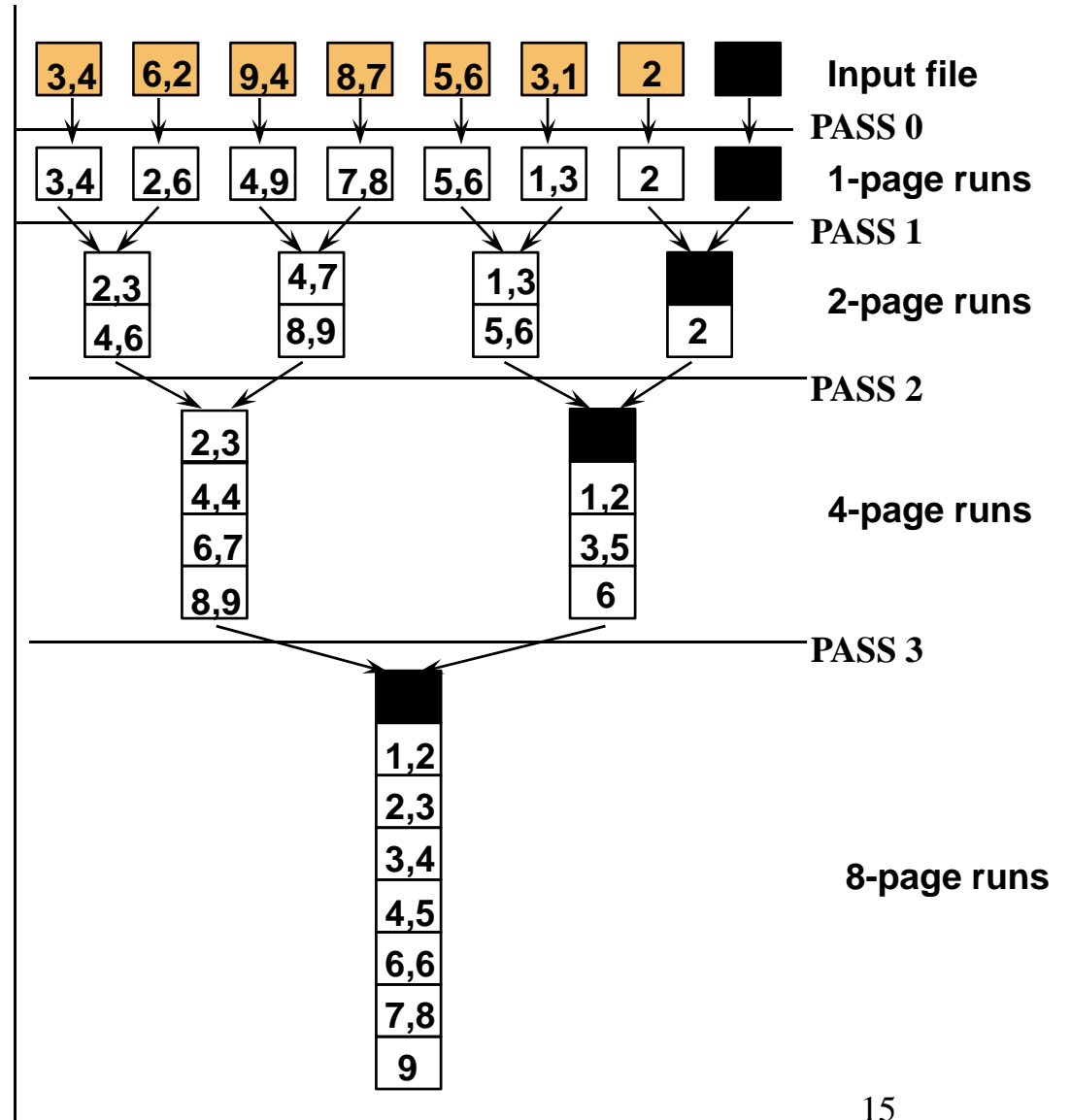


Two-Way External Merge Sort

- Each pass we read + write each page in file.
- N pages in the file => the number of passes

$$= \lceil \log_2 N \rceil + 1$$
- So total cost is:

$$2N(\lceil \log_2 N \rceil + 1)$$
- Sort 4MB with buffer page size = 4KB: needs 11 passes



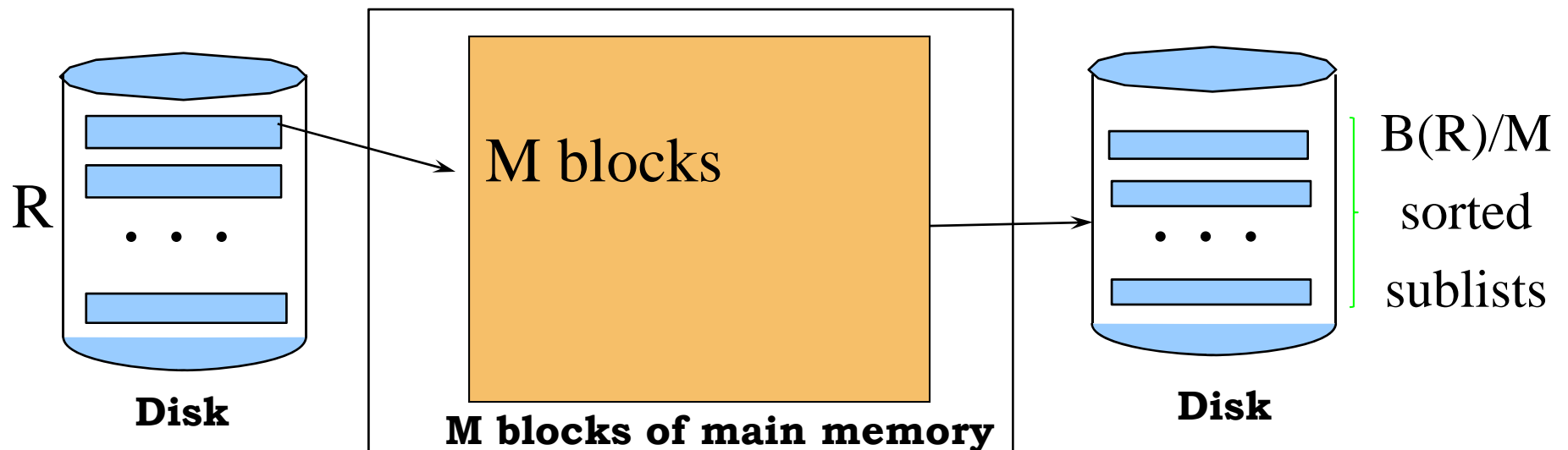
Can We Do Better ?

- We have more main memory
- Should use it to improve performance

- M : # of blocks (i.e., pages) in main memory
- $B(R)$: # of blocks of relation R

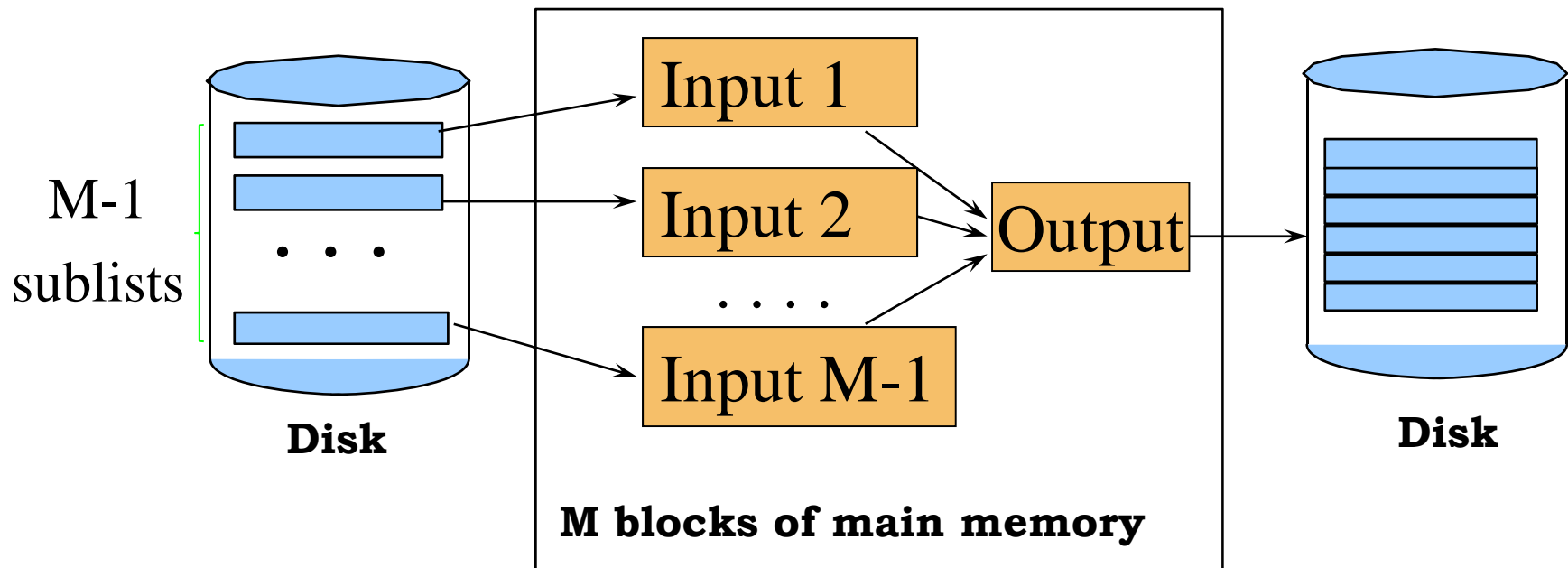
External Merge-Sort

- Pass 0: load M blocks in memory, sort
 - Result: $B(R)/M$ sorted sublists of size M
 - Each sorted sublist is a run



Pass One

- Merge $M - 1$ runs into a new run
- Result: each run has now $M (M - 1)$ blocks

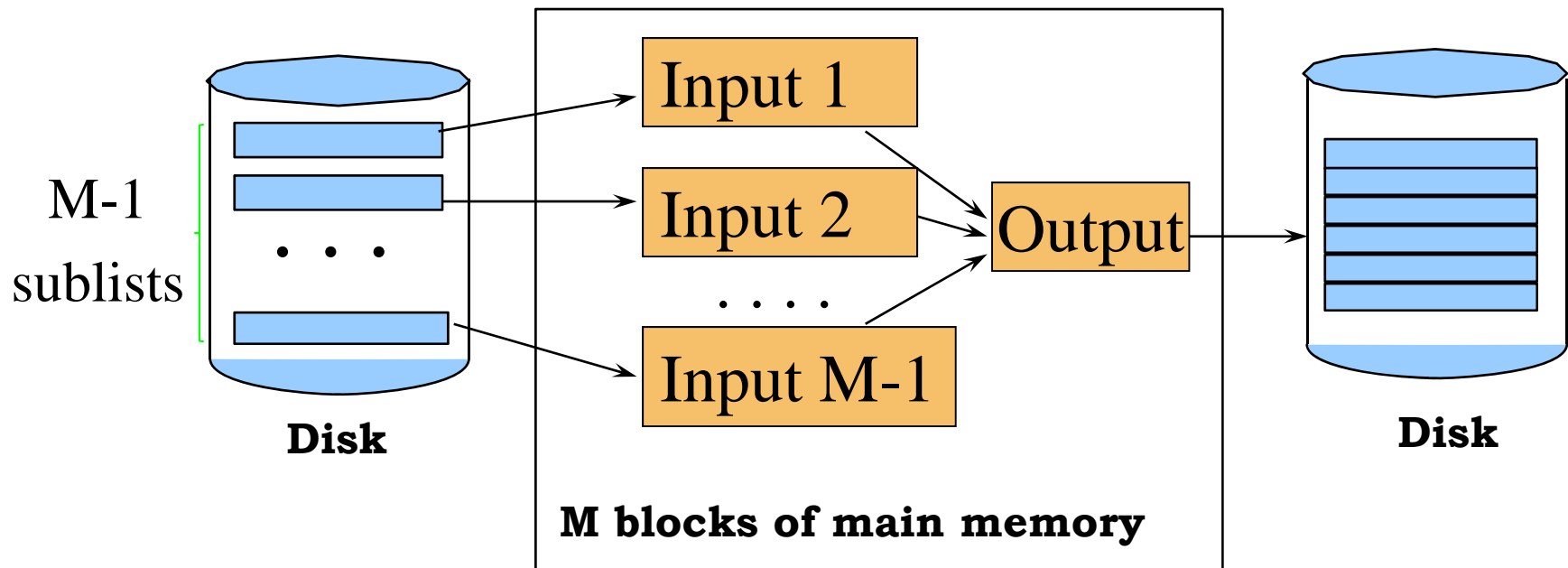


Cost of Two-Pass, Multiway Merge Sort

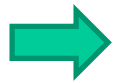
- Pass 0: sort B/M sublists of size M , write
 - Cost: $2B(R)$
- Pass 1: merge B/M sublists, write
 - Cost: $2B(R)$
- Total cost: $4B(R)$
- Assumption: $B(R) \leq M^2$
 - $B/M \leq M - 1$ or
 - $B \leq M(M-1) \sim M^2$

Generalized to k Passs

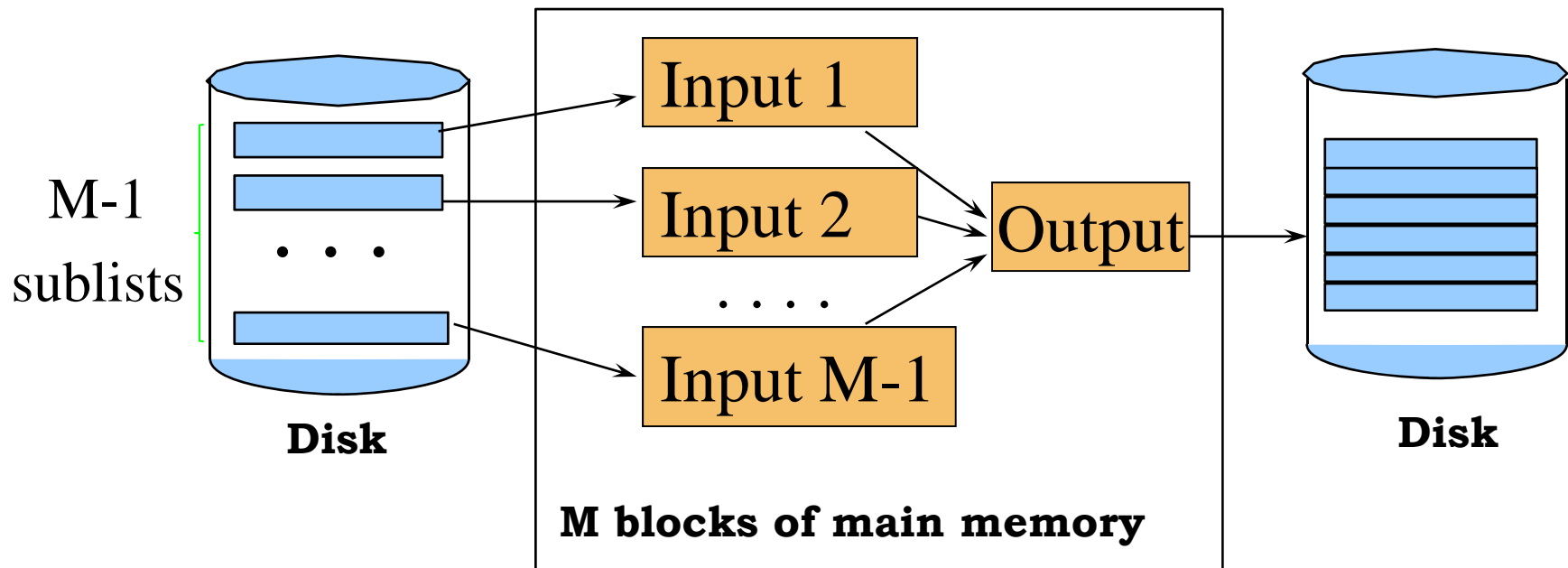
- Merge every $M - 1$ runs into a new run
- Result: each run has now $M (M - 1)^k$ blocks



If k is the last pass

- Merge $M - 1$ runs into a single run
- We must have $M (M - 1)^k \geq B$ 

$$k = \lceil \log_{M-1} \lceil B / M \rceil \rceil$$



Cost of External Merge Sort

- Number of passes: $1 + \lceil \log_{M-1} \lceil B / M \rceil \rceil$
- Cost = $2B * (\text{\# of passes})$
- E.g., with 5 buffer pages, to sort 108 page file:
 - Pass 0: produces $\lceil 108/5 \rceil = 22$ runs (21 sorted runs of 5 pages each + last run of only 3 pages)
 - Pass 1: $\lceil 22/4 \rceil = 6$ (5 sorted runs of 20 pages each + last run of only 8 pages)
 - Pass 2: 2 sorted runs, 80 pages and 28 pages
 - Pass 3: Sorted file of 108 pages