

## COS 226 Lecture II: External sort/search

### HUGE FILE:

- 1970: 1 million bytes
- 2000: 1 trillion bytes

### Use high-level abstraction

- local memory
  - random access
  - faster sequential access
- external memory
  - far slower than local memory
  - read/write in large blocks
  - faster sequential access
  - restrictions on access

### HUGE FILE:

- thousands of times the size of local memory

**Problem:** Sorting and Searching for huge files

## External sort/search algorithms

### SORTING

- N: number of records
- M: size of memory
- $2P$ : number of external devices

**algorithms:** balanced merge, polyphase merge

### SEARCHING

- N: number of records
- M: page size

**algorithms:** indexed sequential, B-trees, extendible hashing

Many ancient algorithms still relevant

## Balanced multiway merge

Make multiple passes over the file

- can MERGE huge sorted blocks in memory
- use half the devices for "output"

pass 0:

- sort the file into sorted blocks of size  $M$   
on devices  $0, 1, 2, \dots, P-1$

pass 1: make blocks of size  $M \cdot P$

- $P$ -way merge out to devices  $P, P+1, P+2, \dots, 2 \cdot P$

pass 2: make blocks of size  $M \cdot P^2$

- $P$ -way merge out to devices  $0, 1, 2, \dots, P-1$

...

pass  $t$ : make blocks of size  $M \cdot P^t$

File is sorted when  $M \cdot P^t > N$

- $t = \log_P (N/M)$  passes through the data

## Balanced merge analysis

Can actually make initial runs of size  $2M$  (use a PQ)

.	PC sort	workstation
. file	1 billion	1 trillion
. memory	1 million	1 billion
. devices	4	10
. passes	$\lg 500 = 9$	$\log_5 500 = 5$

### Costs

- number of passes (no. times each datum touched)
- number of devices

### Bottom line:

Can sort a file in 5-10 times the amount  
of time it takes to read or write it

**IF** enough devices are available

## Balanced merge example

input

T H E Q U I C K B R O W N F X J M P S V L A Z Y D G

sorted runs of size 3

E H T **I Q U** B C K O R W F N X J M P L S V A Y Z D G \*

sorted runs of size 9

B C E H I K Q T U **F J M N O P R W X** A D G L S V Y Z \*

output

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z \*

sorted runs (lengths) on devices      merge cost

	0	1	2	3	4	5	
.	-----						
.	3(3)	3(3)	3(3)	0	0	0	
.	0	0	0	1(9)	1(9)	1(9)	27
.	1(27)	0	0	0	0	0	27

## Polyphase Merge

### Concept:

- cut the number of devices needed in half

### MERGE-UNTIL-EMPTY

- one device empty
- different number of runs  
on other devices
- merge to empty device  
until another device is empty
- iterate

Strategically place runs on devices such that

- last merge empties all but one device to one final run

## Polyphase merge run placement

Work backwards to figure placement

Ex: 4 devices

.	1	0	0	0
.	0	1	1	1
.	1	0	2	2
.	3	2	0	4
.	7	6	4	0
.	0	13	11	7
.	13	0	24	20

distribution for 57 runs

General pattern is complicated!

## Polyphase merge example

sorted runs of size 3

. 0      0      0      1      1      3      3      3      3  
E H T I Q U B C K O R W F N X J M P L S V A Y Z D G \*  
result of phase 1