27/11-12

ALGORITHMS FOR MASSIVE DATA

RASMUS PAGH

OUTLINE

- HARDWARE BACKGROUND, I/O MODEL, PARALLEL DISK MODEL.
- EXTERNAL SEARCH (B-TREES) [Sanders03]
- EXTERNAL SORTING: [Vitter08]
  - MERGE SORT ($D=2$ DISKS)
  - SIMPLE RANDOMIZED MERGE SORT
  - LOWER BOUNDS [Sanders03]
- PROCESSING MASSIVE DATA THROUGH SORTING
  - MAPREDUCE PROGRAMMING MODEL [Lin Dyer 10]
  - EXAMPLES: TRIANGLE LISTING, (CONNECTED COMPONENTS), INVERTED LISTS

EXTERNAL SEARCH

BINARY SEARCH TREE: SPLIT KEY SET IN TWO PARTS OF SIMILAR SIZE

B-TREE: SPLIT KEY SET IN B PARTS OF SIMILAR SIZE

$\Rightarrow$ SEARCH TIME t SATISFIES $B^t \leq N$ (ASSUMING EVEN SPLIT)

$\Rightarrow t \leq \log_B N$.

OPTIMAL IF WE ACCESS DATA ONLY THROUGH POINTERS. (WHY?)

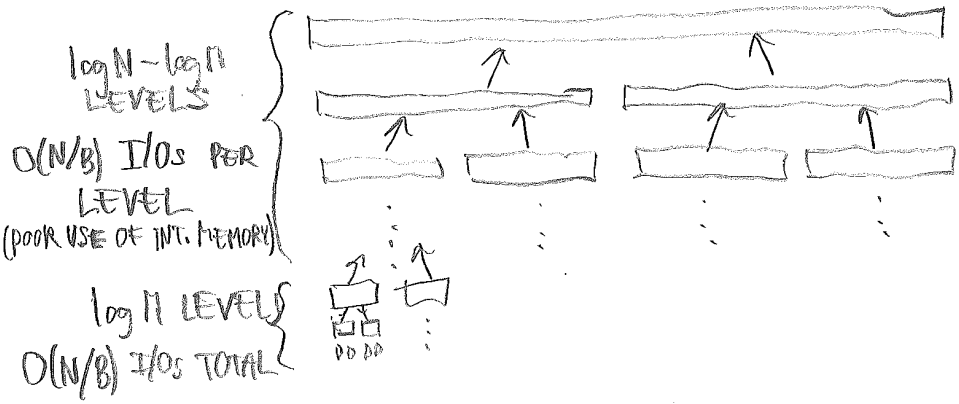
ANSWER: IN t STEPS CAN READ AT MOST B^{t+1} BLOCKS.

IF $t < \log_B N + 1$ THERE IS SOME DATA ELEMENT THAT CANNOT BE FOUND.

29/11-12

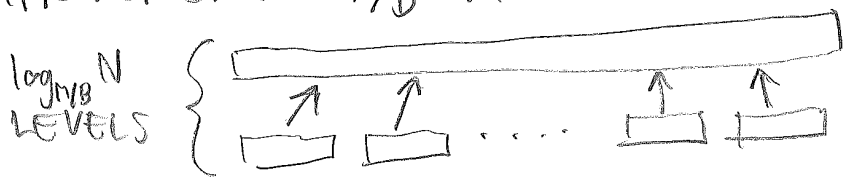
EXTERNAL SORTING

STANDARD (BINARY) MERGE SORT ON DISK (WITH LRU PAGING):



TOTAL I/O COST: $O(\frac{N}{B} \log(N/M))$.

IMPROVEMENT: M/B-WAY MERGING

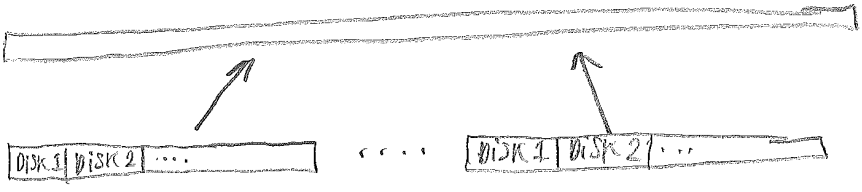


TOTAL I/O COST: $O(\frac{N}{B} \log_{M/B}(N/M))$.

PARALLEL DISK VERSION:

VERSION 1: USE STRIPING TO SIMULATE ONE DISK W. BLOCK SIZE DB.
DISADVANTAGE: CAN DO ONLY $\frac{M}{DB}$-WAY MERGING

VERSION 2: SIMPLE RANDOMIZED MERGE SORT



IDEA: RANDOMIZE START DISK FOR EACH RUN SO
(SKETCH) THAT THE LOAD ON EACH DISK IS BALANCED
THROUGHOUT THE MERGE.

27/11-12

LOWER BOUND FOR SORTING (ONE DISK) [SKETCH]

INDIVISIBILITY ASSUMPTION: SORTING IS ACHIEVED BY MOVING KEY TO/FROM DISK BLOCKS.

FACT: TO SORT ANY INPUT SEQUENCE, AN ALGORITHM NEEDS TO BE ABLE TO DO $N!$ DIFFERENT PERMUTATIONS. (WHY?)

OBSERVATION: WHEN WRITING A BLOCK THERE ARE $\binom{M}{B}$ CHOICES.

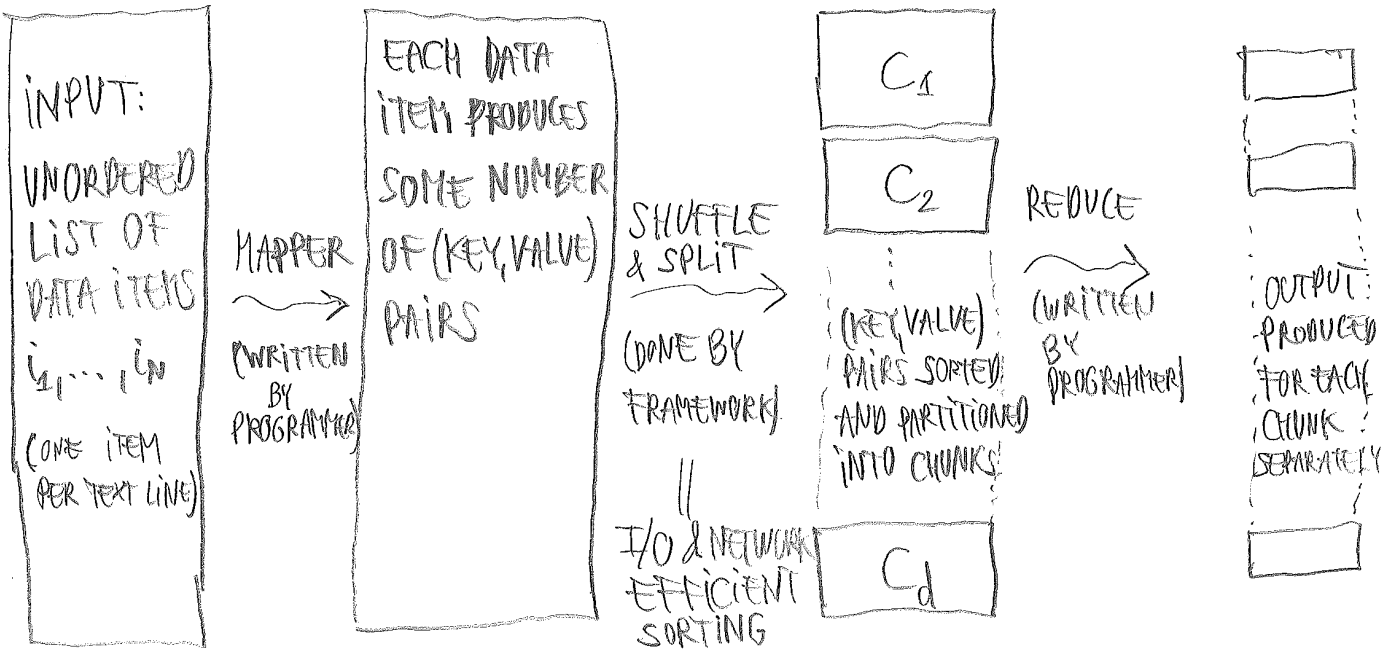
NUMBER OF POSSIBLE PERMUTATIONS AFTER t WRITES

$$\leq \binom{M}{B}^t \approx \left(\frac{M}{B}\right)^{Bt}$$

SO $\left(\frac{M}{B}\right)^{Bt} > N! \Rightarrow Bt \log\left(\frac{M}{B}\right) > \log N! \approx N \log N \Rightarrow t > \frac{N}{B} \log_{M/B} N$

MAPREDUCE PROGRAMMING MODEL

(HADOOP) → AVAILABLE E.G. THROUGH AWS.



27/11-12

MAPREDUCE EXAMPLE: INVERTED LISTS

INPUT: SET OF DOCUMENTS, EACH A SEQUENCE OF WORDS

OUTPUT: FOR EACH WORD, A SEQUENCE OF THE DOCUMENTS IT APPEARS IN.

SIZE GETS BIGGER, BUT STILL LINEAR

MAPPER: doc.txt This is a sample ...

$\rightsquigarrow$ (This, doc.txt) (is, doc.txt) (a, doc.txt) (sample, doc.txt) ...

REDUCER: (sample, doc.txt) (sample, dod.txt) (sample, doe.txt) ... $\rightsquigarrow$

sample: doc.txt, dod.txt, doe.txt ...

MAIN COMPLEXITY MEASURE: AMOUNT OF DATA PRODUCED

MAPREDUCE EXAMPLE: LISTING TRIANGLES

INPUT: EDGES IN UNDIRECTED GRAPH

OUTPUT: LIST OF TRIANGLES IN GRAPH

SIZE PROPORTIONAL TO NUMBER OF SUCH PATHS

MAPPER 1: $\{u,v\} \rightarrow (u,v)\ (v,u)$

REDUCER 1: $(u,v_1)(u,v_2)\ldots(u,v_d) \rightsquigarrow$ $((v_1,v_2),u)\ ((v_1,v_3),u)\ldots((v_{d-1},v_d),u)$ $((u,v_1),-),\ldots,((u,v_d),-)$

PATHS THROUGH u OF LENGTH 2

COPY OF ALL EDGES FROM u AS KEYS, NO ASSOCIATED DATA

MAPPER 2: COPIES OUTPUT FROM FIRST MAPREDUCE PHASE

REDUCER 2: $((u,v),-)\ ((u,v),w_1)\ldots((u,v),w_k) \rightsquigarrow (u,v,w_1),\ldots,(u,v,w_k)$

$((u,v),w_1)\ldots((u,v),w_k) \rightsquigarrow$ NO OUTPUT.

BREADTH-FIRST SEARCH/CONNECTED COMPONENTS CAN BE DONE USING SIMILAR IDEAS. CHALLENGE: SMALL NO. ROUNDS