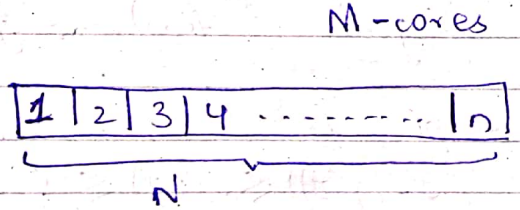
PDC ASSIGNMENT 01

BILAL AHMED KHAN

20K0183 ; Sec:B

QUESTION 01:



Suppose we have "m" number of cores to calculate the average of an N sized 1-D array. We will do the following steps.

1) Divide the array in "m-1" equal parts.

2) Mapp each part on a core from 0, 1 .... m-1 to calculate the sum of the partitioned array

3) The mth core will be reserved for calculating the final sum of the array by adding the local results of each core

4) The last core will the divide the total sum by N to calculate the average of the 1-D array

QUESTION 02:

Pipelining :-

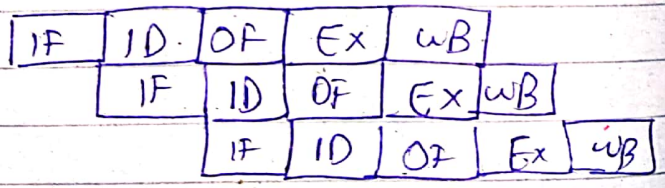
Pipelining is the process of starting a new task before finishing the previous tasks.

A normal instruction goes through the following stages.

1) Instruction Fetch (IF)
2) Instruction Decoding (ID)
3) Operand Fetch (OF)
4) Execution (E)
5) Write-Back (WB)

We schedule our instructions such that every component of the CPU is continously working.

Example:

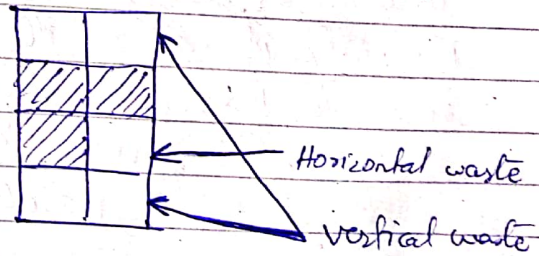


Super Scalar Execution :-

The use of multiple pipelines & the ability of a processor to issue multiple instructions in the same cycle is called superscalar execution.

WASTES in Superscalar Execution :-

The empty slots that occurs while the execution of instructions due to stalls is called "waste"



QUESTION 03:

Cache Mapping:-

Cache mapping refers to the techniques used to transfer data from the main memory to caches.

Mapping Techiques in caches:-

01) Temporal locality:-

It refers to the reuse of a specific data/resource within a specific amount of time. Basically it caches the current data/ resources since it may be needed in the near future so that we don't have to fetch it from the memory.

02) Spatial locality

It is generally refered to the used of data elements within the close proximity of current data.

Spatical locality is useful when accessing linear or sorted data from main memory

QUESTION NO. 04

Given Condition:-

> 8 cores available
> Have to multiply 2 16x16 matrices.

Proposed solution

i) Analysis of Problem:-

> Every row of matrix A will be multiplied by every column of matrix B
> Total No. of row x column multiplication = 16 x 16

ii) Solution:-

> Distribute 2 rows of matrix A to multiply with all colums of matrix B to each core.

> Core 01: will multiply row 1 & 2 of A with all columns of matrix B
Core 02: will multiply row 3 & 4 of Mat A with all colums of matrix B and so on.

iii) Complexity Analysis:-

a. Serial execution:-

Serial execution will result in $O(n^3)$ i.e. $16^3 = 4096$ operations on a single core.

b. Parallel execution:-

will result in $32 \times 16 = 512$ operations per core.

Overall improvement

$$\frac{4096}{512} = 8 \text{ times}$$

Thus our Algorithm is 8 times efficient than serially combining both matrices

QUESTION 05

Tid	Items
00	A, C, D
01	B, C, E, D
02	A, B, C, E
03	B, E, A
04	A, B, C, D, E, F
05	F, B, A
06	A, D, F, E
07	E, C, D, B
08	A, D, F, C
09	A, C, E, B
10	B, D, C, A, F
11	A, C, E, B
12	B, C, E, D, A
13	C, E, D, B, F
14	A, B, F, C
15	D, E, C, F
16	A, B, C, D
17	B, C, F
18	A, B, C
19	D, E, F

PROBLEM BREAKDOWN:-

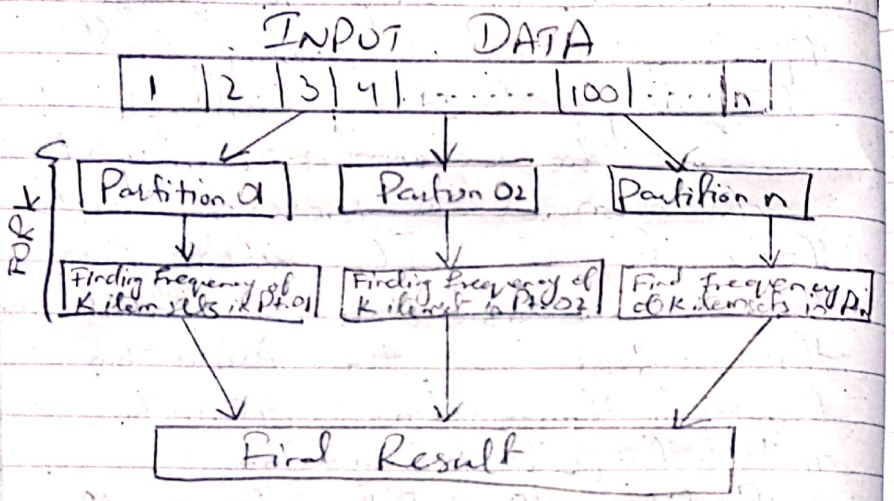
> The key step in ~~Apriori~~ Apriori Algorithm is finding the frequency of each item set.

> The occurence counting of each item set is irrelevent with the counting of another itemset in the transactions. We can parallelize this segment of the Algorithm by using the following algorithm.

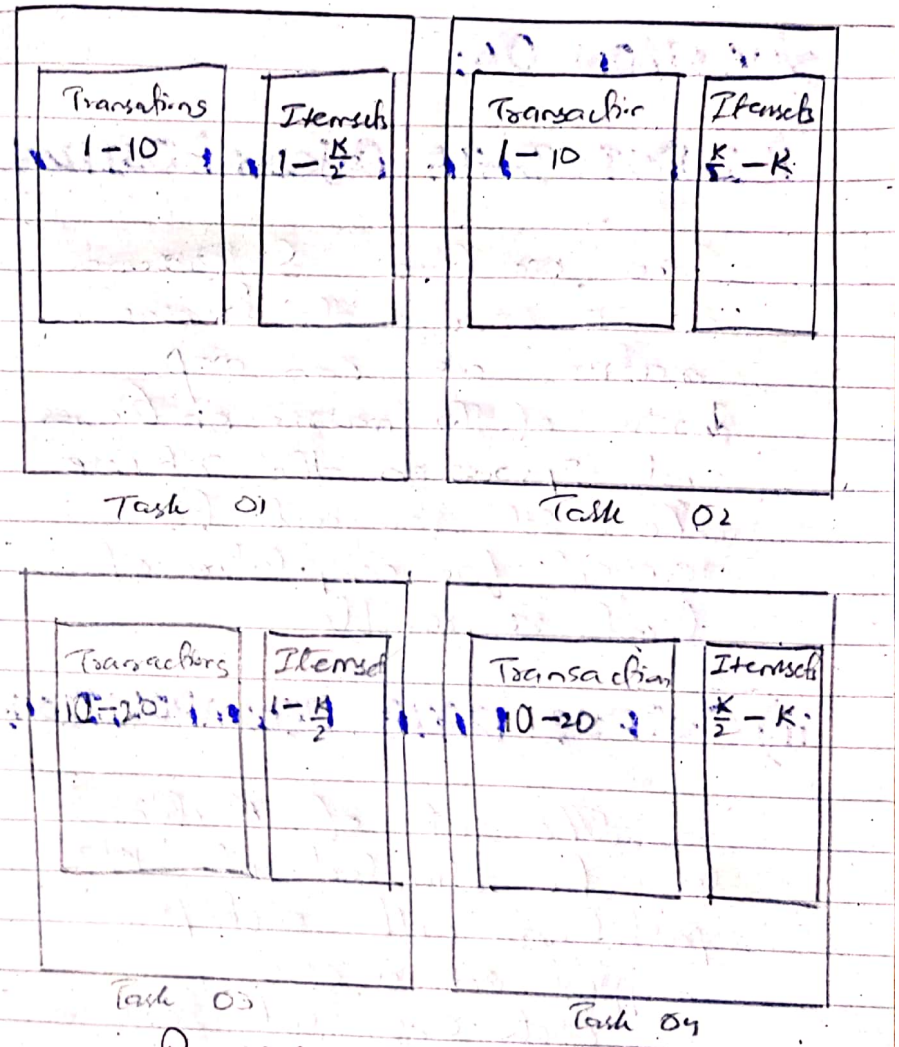
PROPOSED ALGORITHM:-

01) Find frequent 1-itemsets.
02) Partition the data into subdata partitions
03) Find the local frequencies of itemsets on each core
04) Combine the local results to form global results.

The algorithm can be pictorially represented by:-



The actual parallel task distribution is demostrated using diagrams on the next page:



Parallel Execution of Apriori's Algorithm.

QUESTION 06:

i) INPUT DATA DECOMPOSITION

Since we have 8 cores & 20 rows in the transaction matrix we can map 3 rows of the matrix on 6 cores and 2 rows on the 7th core. The 8th row will be reserved for compilation of local results.

ii) OUTPUT DATA DECOMPOSITION

The set of K itemsets can be divided into K/7 partitions with each partition mapped on a core.

Each core will traverse through the transition table, and calculate its delegated K-itemsets frequency. In the end the reserved 8th core will compile the local results to form global results

iii) INTERMEDIATE DATA DECOMPOSITION:

Both datasets i.e. The transaction table, as well as the k-item sets will be divided equally upon the 7 cores. The 8th reserved core will compile the local results at the end to produce global result.