20/01/20

Models of Computation

Both sequential & parallel computer operates on a set of instruction called algorithms. These set of instructn instructs the computer about what it has to do in each step.

Depending on instruction stream & data stream comp can be classified into 4 categories.

SISD (Single instructn single Data stream)

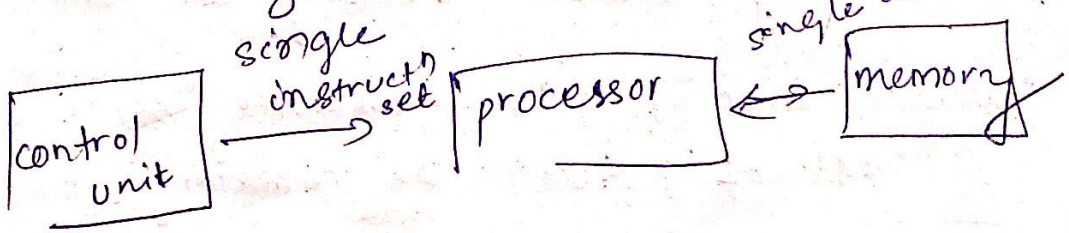
SIMD (" " multiple

MISD (multiple single

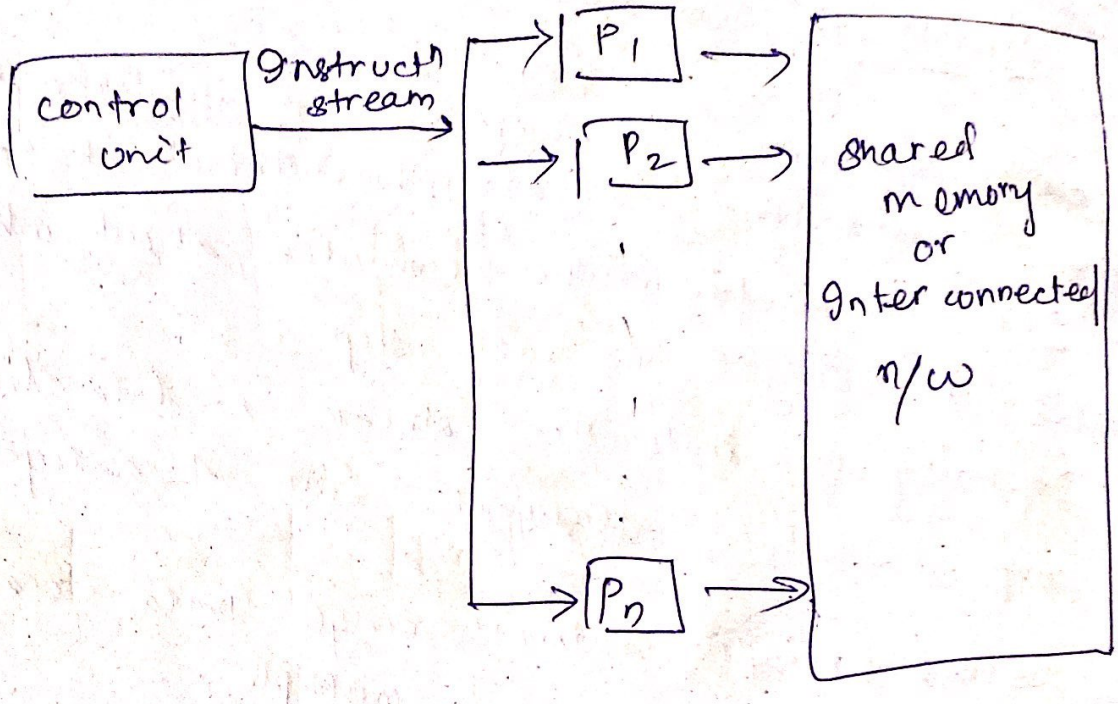
MIMD

SISD - In this type of computer single processor can receive instructn stream from control unit & again it operates on a single data stream.

During computation at each step, the processor receives one instructn from control unit & operates on a single data received from the memory unit.



SIMD



SIMD contains

During computat at each step, all processors receive a single instruction set from the control unit & operates on diff set of data.

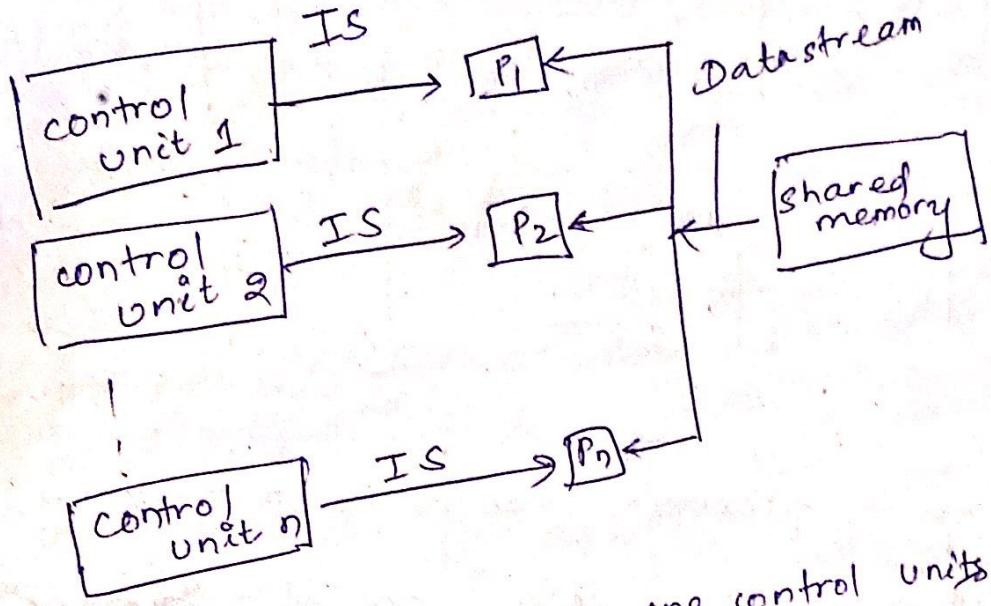
Each of the processing unit have its own local memory to store data & instruct.

In SIMD comp., the processor need to communicate among themselves. This is done by shared memory or Interconnect n/w.

While some of the processors will execute a set of instruct, the remaining processor wait for their next set of instruct.

Instruction from control unit decides which processor will be active or inactive

MISD -



It consists of more than one control units & multiple

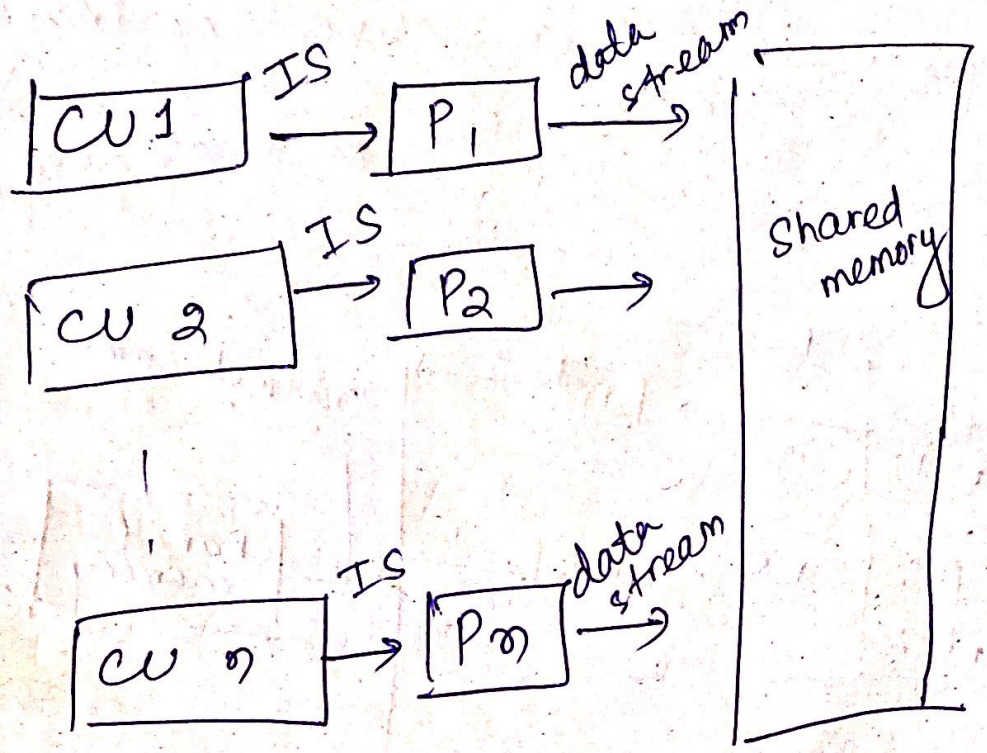
MISD compl.. contains

Each processor has its own control units and they share a common memory units

All the processors gets instructn individually from their own control unit & they operates on a single stream of data. As per the instructn they have received from their respective control unit.

MIMD

MIMD



Each processor has its own memory, contro
und
They receive diff. set of instruct
from their own control unit and
operates on diff. sets of data stream.

- Multicomputer.
- Multidistributed s/m.

PDS

Analysing Parallel algorithm

We can analyse This refers to the process of determining how good an algom is that how fast, & how expensive to run & how efficient it is in its use of available resources.

It is usually evaluated using following criteria -

1. Running time
2. No. of processors used
3. Cost

1. **Running time** - (It is essential to speed of opn.)

It is defined as time taken by the algom to solve a problem on ||el computer i.e the time lapse from the starting of algom to the moment the algom terminates.

counting steps

Before actually implementing the algorithm on a computer it is necessary to conduct a theoritical analysis of the time it will require to solve the computational problem. as executed by the algo^m at worst case. The running time of a ||el algo^m usually obtained by counting 2 kinds of steps-

(i) Computational step
(ii) Routing step

A comp. step is an Arithmatic or logic opr^n perform on a piece of info^m within a processor.

Routing step - The piece of info^m travels from one processor to another processor via shared memory or interconnect^n n/w.

Speed up is eventually a ||el algo^m for a given problem.

$$\text{Speed up} = \frac{\text{worst case running time of a known fastest sequential algo}^m \text{ for a particular problem}}{\text{worst case running time of ||el algo.}}$$

(*) It detects faster the speed up better the ||el algo^n.

Cost

cost of the ||el algom is defined as the product of

cost = ||el running time * no. of processors used

efficiency = coorst case running time of fastest known sequential algom for a problem

cost of ||el algom.

10/02/2020

MATRIX MULTIPLICATION USING PARALLEL PROCESSORS –

single processor -

n^{el} c/m -

Algorithm

procedure MATRIXMULTI

begin

for k=1 to n-1 step 1 do

begin

for all Pij, where i & j ranges from 1 to n do

if i is greater than k

then rotate aij in the east direct?

if j is greater than k then

rotate in soeth direction

end if

end.

Step-2 for all Pij, where i,j lies b/w 1 to n do

compute the product of a and b, and store the result in c.

for k=1 to n-1 step 1 do for all Pij, where i & j ranges from 1 to n

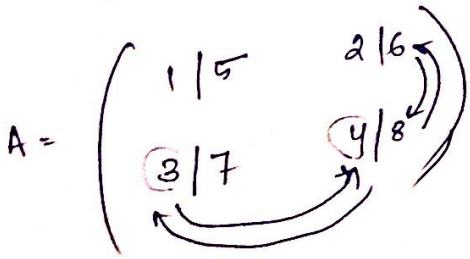
rotate a in east direction

rotate b in south direction

c = c + axb

end

Eg. $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$



processor1 - a_{11}, b_{11}

processor2 - a_{12}, b_{12}

processor3 - a_{21}, b_{21}

processor4 - a_{22}, b_{22}

$n^2 = 4$ (no of processor)

$n = 2$

k = 1 to (2-1) for i.

$a_{21} > k$, $b_{21} > k$

$a_{22} > k$ $b_{22} > k$.

for both i & j

$A = \begin{pmatrix} 1 & 5 & 2 & 8 \\ 4 & 7 & 3 & 6 \end{pmatrix}$ $\Rightarrow A = \begin{pmatrix} 1\times5 & 2\times8 \\ 4\times7 & 3\times6 \end{pmatrix} \Rightarrow A \begin{pmatrix} 5 & 16 \\ 28 & 18 \end{pmatrix}$ — (I)

↳ store the value in memory.

Step 2

$A = \begin{pmatrix} 2 & 7 & 1 & 6 \\ 3 & 5 & 4 & 8 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 14 & 6 \\ 15 & 32 \end{pmatrix}$ — (II)

↳ store the value in memory.

(I) + (II) $\Rightarrow$

$C_{ij} = A = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$ [Matrix multiplicatn using 11 el processer]

Q// $A = \left[\begin{array}{cc|cc} 2 & 1 & 5 & 3 \\ 6 & 7 & 1 & 6 \\ \hline 9 & 2 & 4 & 4 \\ 3 & 6 & 7 & 2 \end{array}\right]_{4\times4}$ $B = \left[\begin{array}{cc|cc} 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 5 \\ \hline 1 & 9 & 8 & -8 \\ 4 & 0 & -8 & 5 \end{array}\right]_{4\times4}$

$P_0 = \begin{bmatrix} 2 & 1 \\ 6 & 7 \end{bmatrix}$ $P_1 = \begin{bmatrix} 5 & 3 \\ 1 & 6 \end{bmatrix}$ $P_2 = \begin{bmatrix} 9 & 2 \\ 3 & 6 \end{bmatrix}$ $P_3 = \begin{bmatrix} 4 & 4 \\ 7 & 2 \end{bmatrix}$

$P_0 = \begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix}$ $P_1 = \begin{bmatrix} 2 & 3 \\ 6 & 5 \end{bmatrix}$ $P_2 = \begin{bmatrix} 1 & 9 \\ 4 & 0 \end{bmatrix}$ $P_3 = \begin{bmatrix} 8 & -8 \\ -8 & 5 \end{bmatrix}$

4 processor
$n^2 = 4$
$n = 2$

$A = \begin{pmatrix} AP_0 \mid BP_0 & AP_1 \mid BP_1 \\ AP_2 \mid BP_2 & AP_3 \mid BP_3 \end{pmatrix} \Rightarrow A = \begin{pmatrix} AP_0 \mid BP_0 & AP_1 \mid BP_3 \\ AP_3 \mid BP_2 & AP_2 \mid BP_1 \end{pmatrix}$

$A = \left(\begin{array}{cc|cc} 2 & 1 & 5 & 3 \\ 6 & 7 & 1 & 6 \\ \hline 4 & 4 & 8 & -8 \\ 7 & 2 & -8 & 5 \end{array}\right) \left(\begin{array}{cc|cc} 6 & 1 & 8 & -8 \\ 4 & 5 & -8 & 5 \\ \hline 1 & 9 & 2 & 3 \\ 4 & 0 & 6 & 5 \end{array}\right) = \left(\begin{array}{cc|cc} 16 & 7 & 16 & -25 \\ 28 & 55 & -40 & 22 \\ \hline 20 & 36 & 30 & 37 \\ 15 & 63 & -42 & 37 \end{array}\right)$

(I) ↳ store the value.

step-2

$$A = \left(\begin{array}{cc|cc} 5 & 3 & 1 & 9 \\ 1 & 6 & 4 & 0 \\ \hline 9 & 2 & 6 & 1 \\ 3 & 6 & 4 & 5 \end{array}\right) \quad \left(\begin{array}{cc|cc} 2 & 1 & 2 & 3 \\ 6 & 7 & 6 & 5 \\ \hline 4 & 4 & 8 & -8 \\ 7 & 2 & -8 & 5 \end{array}\right)$$

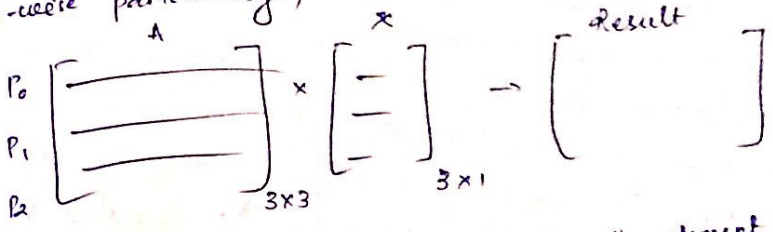
$$A = \left(\quad\right)$$

(1) + (11) =>

12/02/2020

In case of vector, we will assign each row

(One directional
Row-wise partitioning)



* P_0 contains 0^{th} row of vector A & 0^{th} element of vector X. Similarly, P_1 → 1^{st} row of vector.

* for multiplication, P_0, P_1, P_2 will broadcast its single element of X vector to the other processors. This called all to all broadcast.

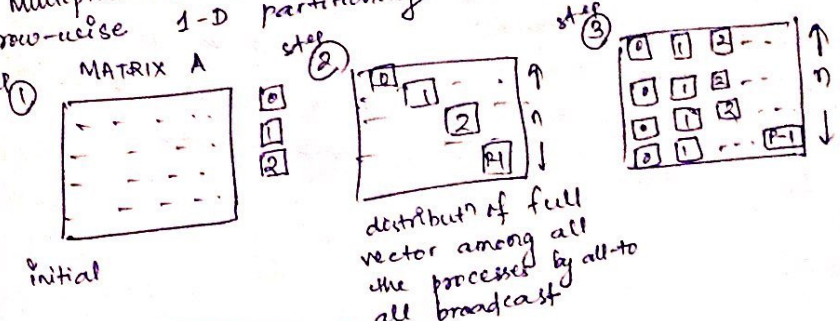
* Vector 'X' is common for all.

After multiplicatn, Resultant matrix is stored in result matrix.

Time complexity - $O(n^2)$

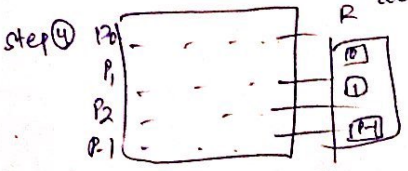
Matrix vector multiplicatn By Using 1-D partitioning

* Multiplication of an nxn matrix with an nx1 vector using row-wise 1-D partitioning for 1 row per process.



initial

distributn of full vector among all the processes by all-to all broadcast



final distributn of all matrix & result vector R

Steps

S#1 Consider the case in which the $n \times n$ matrix is partitioned among 'n' processes stores 1 complete row of the matrix. Then $n \times 1$ vector 'x' is distributed such that each process owns 1 of its element.

S#2 - The initial distribution of the matrix & the vector for row wise block 1-D partitioning.

S#3 - Process 'P_i' initially owns x_j & $A[i,0], A[i,1], A[i,2] \cdots A[i,n-1]$ and is responsible for computing Y_i (Resultant Vector).

S#4 - Vector 'x' is multiplied with each row of the matrix so every process needs the entire vector. Since each process starts with only one element of 'x'. An all-to-all broadcast is required to distribute all the elements to all the processes. After vector 'x' is distributed among all processes process P_i computes $Y[i] = \sum_{j=0}^{n-1} (P[i,j](x) * [j])$.

* The result vector Y is stored, exactly the way the starting vector 'x' was stored.

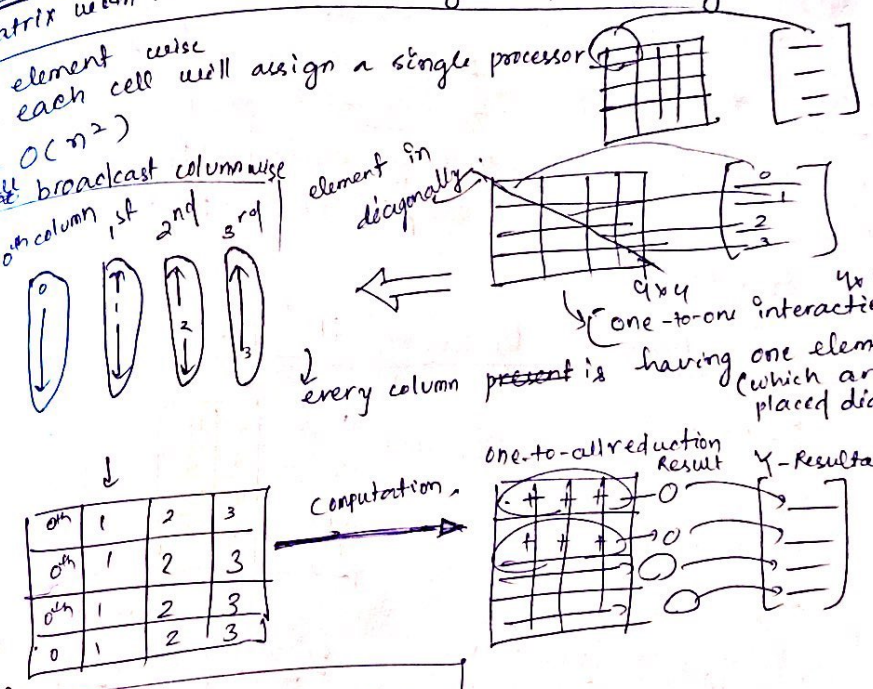
Parallel Run Time - Starting with one vector element per process the all to all broadcast of the vector elements among 'n' processes requires the time $O(n)$ of any architecture.

- The multiplication of a single row of A with 'x' is also perform by each process in time $O(n)$.
- Thus the entire process is completed by 'n' processes in the time of $O(n)$. Resulting in a process time product of $O(n^2)$. The ||el algo is cost optimal.

18/02/2020

Matrix with vector multiplication using 2D-partitioning -

- element wise
- each cell will assign a single processor
- $O(n^2)$

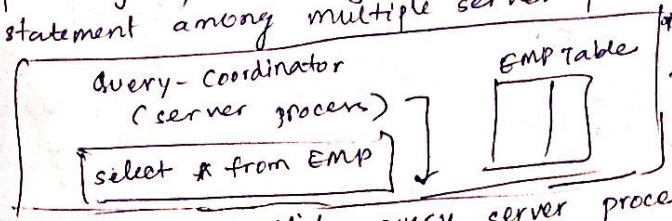


steps

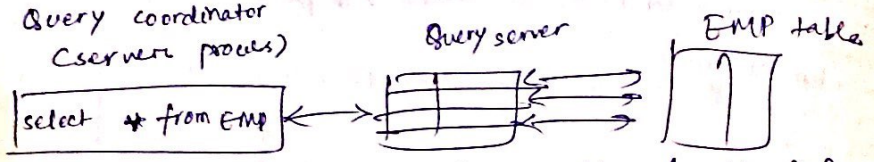
1. one-to-one interaction
2. one-to all broadcast
3. computation
4. one-to-all reduction — accumulate row wise. store the result in new vector

Database Query Processing Using Parallel Computer

* The processing of a SQL statement is always perform by a single server process with the ||el query features multiple processes can work together simultaneously to process a single SQL statement. This capability is called ||el query processing. By dividing the work necessary to process a statement among multiple server processes.

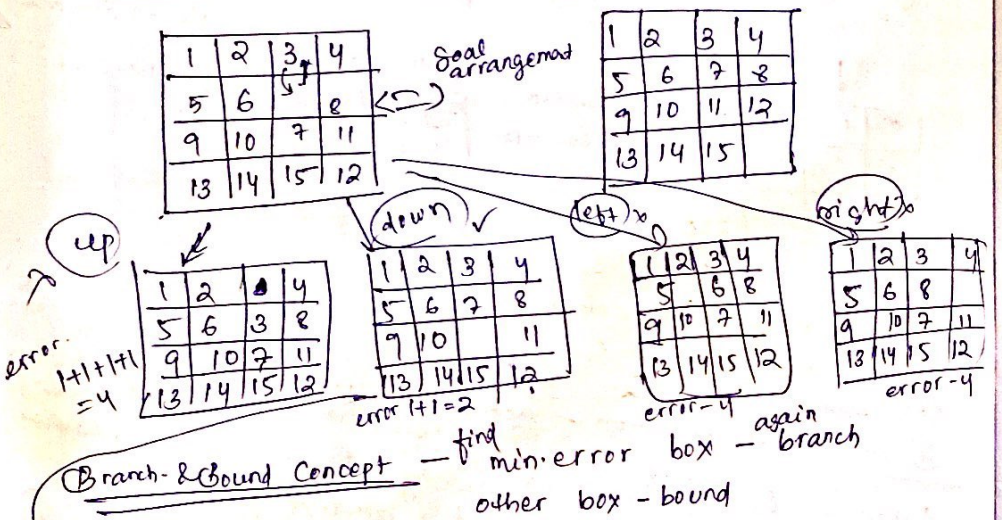


* The parallel query feature allows certain operations to be perform in parallel by multiple query server processes, dispatches the execution of statement to several query servers & coordinates the results from all of the servers to send back to the user.

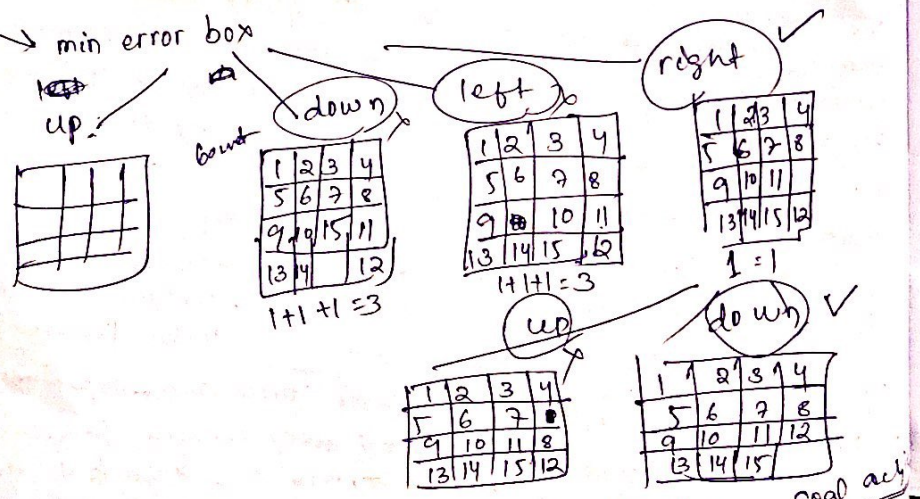


Query coordinator can breakdown the execution function into pieces & then integrates the partial result produced by the Query server.

15 puzzle problem by using n/w processors –



Branch-&-Bound Concept – find min. error box – again branch other box - bound



19/02/2020

Sorting –

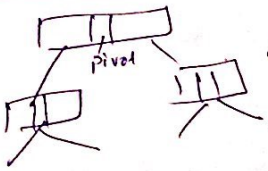
Sorting is a process of arranging elements in a group in a particular order i.e in ascending or in descending order.

Diff. types of techniques –
1) Enumeration sort
2) Odd-even sort
3) parallel merge sort
4) Hyper quick sort
5) Bitonic merge sort

1) Enumeration sort

Enumeration sort is a method of arranging all elements in a list by finding the final position of each element in a sorted list. It is done by comparing each element with all other element & finding the no. of element having smaller value.

2) ~~Odd-even~~ Quick sort



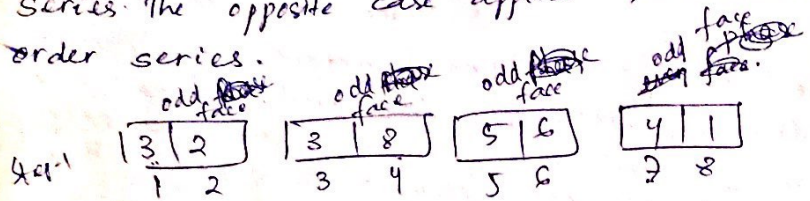
* Quicksort is not cost optimal in case of parallel processes.
* If we assign no. of processor for division process of particular array, then quicksort is cost optimal

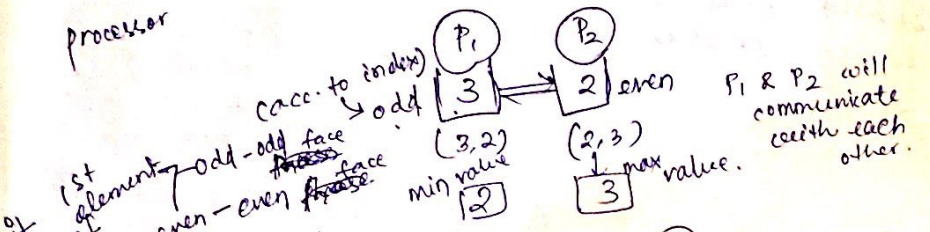
4) Hyper quick sort is an implementation of quick sort & hyper queue. Its steps are –

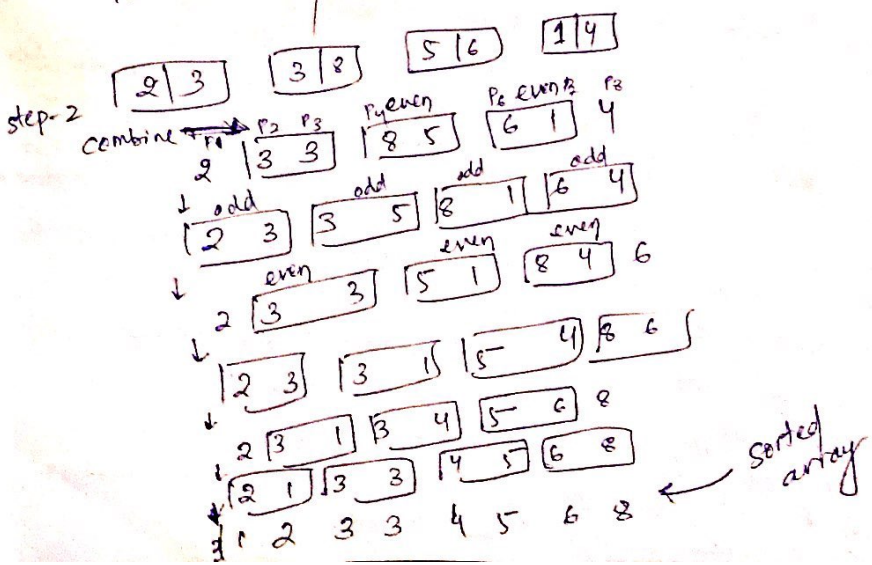
1) divide the unsorted list among each node
2) sort each node locally from node '0'
3) broadcast the median value
4) split each list locally, then exchange the halves across the highest dimension.

5) Repeat 3 & 4 in 1[e] until the dimension reaches to zero.

2) Odd-even-Transposition sort - It is based on the bubble sort technique. It compares 2 adjacent nos & switches the if the 1st no. > 2nd no. to get an ascending order series. The opposite case applies for a descending order series.







1[e] Merge sort - divide & conquer

It 1st divides the unsorted list.

It just follow the parallelism principle.

Bitonic merge sot - It is a procedure of sorting a bitonic sequence using bitonic splits. A sequence is said to be bitonic if it gradually increases; after half position, it gradually decreases.

* An array given below. is bitonic if there exist an index 'i' where arr[0, ... n-1]

$0 <= i <= n-1$ such that

$x_0 <= x_1 \ldots <= x_i$ and

$x_i >= x_{i+1} \ldots >= x_{n-1}$