1.

1.1 Round Robin

→ int i=0

→ for all values of i

→ if(i==0)

→ Send the self value to the neighbours i.e. p_{i-1} and p_{i+1}. (if last node i.e. $p = p-1$ and then only send to its $p-1$).

→ Increment i (i++)

→ else

→ Receive data from the neighbours i.e. $p+1$ and $p-1$. (if last node i.e. $p = p-1$ then only receive from $p-1$).

→ Calculate heat from the equation.

→ Send the value to the neighbours (i.e. $p+1$ and $p-1$)

→ increment i (i++)

* Per iteration of the Heat equation for a Round Robin decomposition, every node after computing the heat equation sends the results to two of its neighbours (i.e. $p-1$ and $p+1$). Only the last node i.e. $p = p-1$ sends the data to its previous node.

∴ Total communication will be 2 × No. of nodes − 1

∴ Communication per node = $\frac{n}{p} \times 2$

Total communication per link = 4

1.2 Block

→ Init i=0

→ for all values of i

→ if (i==0)

→ Send the first and the last element of data allocated to that processor to its neighbours i.e. p-1 and p+1

for p=0, send to p+1

→ Increment i (i++)

→ else

→ Receive data from the neighbours i.e. p+1 and p-1.

for p=0, receive from p+1

for p=p-1, receive from p-1

→ Calculate heat from the equation.

→ Send the first and last element to p-1 and p+1.

for p=0, send to p+1

→ Increment i (i++)

* Communication per iteration is 2 except for the first and the last node.

1.3 Solution

I would use the Block decomposition as the communication is easy as compared to the round robin decomposition.

(2)

2.1. Horizontal

→ Every processor gets n/p rows of A and n/p elements of x. Hence each processor has the partial copy of x. Send the value of x to all the processors.

→ for $i = p * n/p$ to $(p+1) * n/p$

→ Send x[i] to every processor (We can use scatter function)

→ for $i = 0$ to $p * n/p$ and $i = (p+1) * n/p$ to n

→ Receive x[i] from every processor (We can use gather function)

→ Computer multiplication between the n/p rows ($p * n/p$ to $(p+1) * n/p$) for each processor and save the result in y.

→ Copy result in x

→ Repeat for 10 iterations.

* Memory needed = $n^2/p + n$

* Communication per node = p
Communication per link = $p-1$
Total communication = $p * (p-1) * n/p$

2.2. Vertical

→ Every processor will have n/p columns of A. p=0 will have full n.

→ if(p==0)
  → Send n to every processor such that every processor has n/p blocks (Use scatter)

→ else
  → Receive value of n

→ for row 1 to n
  → for column = p * n/p to (p+1) * n/p
    → a[row][col] * n[col]. Store it in temp vector.

→ for i=1 to n and i!=p
  → Send temp vector to i

→ for i=1 to n and i!=p
  → Receive temp vector from other processors and aggregate to form y.

→ Copy ~~and~~ the value of y into n and iterate.

* Memory needed = $n^2/p + n$

* Communication per node = p
Communication per link = p - 1
Total communication $p * (p-1) * n/p$

2.3 Block

→ Every processor will have $n/\sqrt{p} * n/\sqrt{p}$ block of A and $n/\sqrt{p}$ rows of x.

→ Every processor will calculate $n/\sqrt{p}$ rows of y.

→ for row = $p\%\sqrt{p} * n/\sqrt{p}$ to $(p+1)\%\sqrt{p} * n/\sqrt{p}$

→ for col = $p\%\sqrt{p} * n/\sqrt{p}$ to $(p+1)\%\sqrt{p} * n/\sqrt{p}$

→ y[row] = y[row] + a[row][col] * x[col]

→ If p is a non-diagonal element

→ Send y to closest diagonal

→ Diagonals will receive y and aggregate.

→ Copy y into x and send x to all processors in the same row.

→ Repeat for 10 iterations.

* Memory needed = $n^2/p + n$

~~* Communication per node = $(\sqrt{p}-1) * n/\sqrt{p}$~~

* Communication per link = $n/\sqrt{p}$

Total Communication = $(p - \sqrt{p}) * n/\sqrt{p}$

(3)

* Reduce star on chain

Most loaded link = $O(P-1)$

Most loaded node = $O(P-1)$

longest chain of communication = P

* Reduce star on clique

Most loaded link = $O(1)$

Most loaded node = $O(P-1)$

longest chain of communication = 1

* Reduce chain on chain

Most loaded link = $O(1)$

Most loaded node = $O(1)$

longest chain of communication = P

* Reduce chain on clique

Most loaded link = O(1)

Most loaded node = O(1)

longest chain of communication = P

* Reduce tree on chain

Most loaded link = O(P/2)

Most loaded node = O(log P)

longest chain of communication = P/2

* Reduce tree on clique

Most loaded link = O(1)

Most loaded node = O(log P)

longest chain of communication = 1