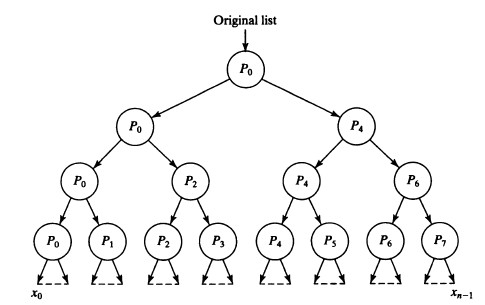
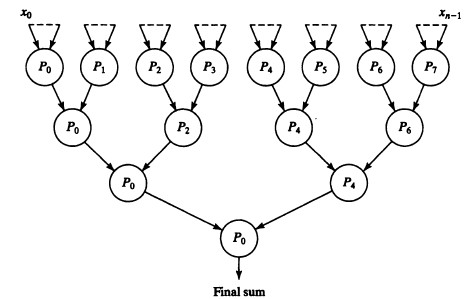
Assignment 2

Due on Monday, April 1, 2019

1. Complete the parallel pseudocode given in the following divide-and-conquer method for all eight processors.





void divide(int \*s, \*s1, \*s2)

{

num = Number(s); // get the length of s array

s1 = s[0:num/2-1];

s2 = s[num/2:num-1]

}

int add(int \*s, \*s1, \*s2)

{

if (number(s)<= 2) return (n1+n2); // base condition, return the sum of 2 elements

else{

divide(s, s1, s2); // divide the array

part\_sum1 = add(s1); // recursive call

part\_sum2 = add(s2);

return (part\_sum1 + part\_sum2);

}

}

void Main()

{

initialize \*s, \*s1, \*s2

produce s array; // obtain the array need to be sum, from file or produce random

for(i=log2(n-1); i<=0; i--){

if (myid & (2i-1) == 0){

if (myid & (2(i+1)-1) == 0){

divide (s, s1, s2);

send s2 to (myid ^ 2i);

if (i=0) partsum = add(s1);

}else{

Receive s1 from (myid ^ 2i)

if (i=0) partsum = add(s1);

}

}

}

for (i=0; i<=log2n; i++){

if (myid & (2i-1) == 0){

if (myid & (2(i+1)-1) == 0){

receive localsum from (myid^2i);

partsum = partsum + localsum;

}else{

send partsum to (myid^2i);

}

}

}

if (myid == 0) printf(“The sum of array is: %d”, partsum);

}

2. Develop a divide-and-conquer algorithm that finds the smallest value in a set of *n* values in O(log*n*) steps using *n*/2 processors. What is the time complexity?

int min(int \*s)

{

if (number(s)<= 2) { // base condition, return the smaller of 2 elements

if ( n1> n2) return n2;

else return n1;

}

else{

divide(s, s1, s2); // divide the array

part\_min1 = min(s1); // recursive call

part\_min2 = min(s2);

if ( part\_min1 > part\_min2) return part\_min2;

else return part\_min1;

}

}

tcomm1 = (p-1) \* tstartup + ntdata((p-1)/p) = (p-1)( tstartup + n/ptdata)

tcomm2 = (p-1) \* tstartup + tdata(p-1) = (p-1)( tstartup + tdata)

tcomp1 = n/p – 1

tcomp2 = p – 1

3. Design an algorithm to find the sum of *n* numbers in parallel using 2-D mesh interconnection work (Assume *p* processors are available, and the original list is on Processor 0). Write the pseudocode, and then analyze the execution time (including communication time and computation time), and calculate the speedup. Please draw the picture to explain the algorithm as well if you think it is necessary.

if ( myid == 0 ){

initialize arr;

}

// For first row, send and receive data

If ( myid% == 0 ){

If ( !empty(arr)){

Send arr[/(p-myid): ] to ( myid + );

arr = arr [ : (/(p-myid) -1)];

}else{

Receive arr from ( myid - );

}

}

// For middle rows, send and receive data

If ( myid % != 0 & ( myid + 1 ) % != 0 ){

If ( !empty(arr)){

Send arr[1/(-myid): ] to ( myid + 1 );

arr = arr [ : (1/(-myid) -1)];

}else{

Receive arr from ( myid - 1);

}

}

// For bottom row, receive data

If ( (myid+1) % == 0 ){

Receive arr from (myid -1 );

}

If ( !empty( arr ) ) localsum = add (arr);

// Collect data from other processors

If ( (myid+1) % == 0 ){ // Bottom row

Send localsum to (myid - 1);

}

If ((myid+1) % != 0 & myid% != 0 ){ // Middle rows

If ( partsum != NULL ){

Send partsum to ( myid -1 );

}else{

Receive partsum from ( myid + 1 );

Partsum += localsum;

}

}

If( myid% == 0 ){

If( myid == p-){

If ( partsum!=NULL ) send partsum to (myid - );

Else receive partsum from ( myid + 1 ) ;

}else{

If ( partsum!=NULL ) send partsum to (myid - );

Else{

Receive temp from ( myid + 1);

Localsum+=temp;

Receive partsum from ( myid + );

Partsum+= localsum;

}

}

}

If ( myid == 0 ) printf (“The total sum is: %d”, partsum);

Analysis:

ts = n-1

tcomm1 = 2( – 1 ) tstartup + ntdata ( ( -1 )/ + ( – 2 )/ + … + 1/ + ( – 1)/P + ( - 2 )/P + …+ 1/P ) = 2 ( - 1) tstartup + ntdata ( ( – 1 )/2 + ( – 1 )/2P )

tcomm2 = 2( – 1 ) ( tstartup + tdata )

tcomp = n/P - 1 + 2 ( – 1 )

Sp = ts/tp = ( n – 1 )/ ( 2 ( - 1) tstartup + ntdata ( ( – 1 )/2 + ( – 1 )/2P ) + 2( – 1 ) ( tstartup + tdata ) + n/P - 1 + 2 ( – 1 ))

4. Assume a sequence of 18 integers 11, 29, 21, 4, 43, 24, 52, 8, 17, 50, 45, 30, 15, 34, 54, 2, 35, 61. Write the detailed steps to sort the integers using bucket sorting (3 buckets with equal size). If there are three processors available, write the detailed steps to sort them using the parallel bucket sorting algorithm (further parallelization version) described in the textbook.