CSCI 455- Parallel Computing

DJG

Assignment #2

Darwin J. Grosklag
200903596

Q. #1 Dev a div & conquer Algo to find smallest value in $O(\log n)$ w $n/2$ processors.

```
void parmergesort(int* A, int start, int n) {
    int rank = getrank(WORLD);
    if (n == 1) return;
    if (rank != start || rank != (start+n)/2 ) return;
    if (rank == start)          // if odd take right

        parmergesort(A, start, n/2);

    }
    else {
        parmergesort(A, start + n/2, n - n/2);

    }
    Barrier();
    if (rank == start)
        merge(A, n);

}
```

Q. 4-11 Ch 4 - Div & Conquer

write pgm to sum n ints & compare performance. let $n = 2^p$ s.t. $p \in \mathbb{Z}$.

Summary: a) n int in $n/2$ pairs (procs)

b) n ints into $\frac{n}{\log n}$ groups ($\log n$ each)

4-11

a)
```
int sum_a(int* A, int start, int n) {
    if (n < 1) return 0;
    if (n == 1) return A[start];
    if (n == 2) return A[start] + A[start+1];

                // calls fork
    t = async_proc(sum_a, {A, start-n/2, n-n/2});
    t.begin();

    int accum = sum_a(A, start, n/2);
    while (! t.complete()) {}

    accum += t.result();
    return accum;
}
```

b)
```
int sum_b(int * A, int n) {
    int temp[n];

    for i=0 to n-1 do in parallel:
        temp[i] = A[i];

    for i=0 to n-1 do in parallel:
        for j=1 to log(2, n) do:
            temp[i] = temp[i] + temp[i - pow(2, j-1)];

    return temp[n-1];

}
```

Assig 2

Q 4-20

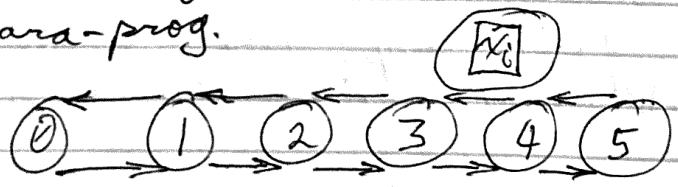
```
int area=0, section=0, start=0, end=0, part_area;
MPI_Init(&argc, &argv);
MPI_Comm_Size (MPI_COMM_WORLD, &np);
MPI_Comm_Rank (MPI_COMM_WORLD, &rank);

if (rank == 0) {
    cout << "Intervals n = ";
    cin >> n;        cout << end;
    if (n % 2 != 0) {
        cout << "n must be even, setting to "
             << ++n
             << ".\n"
    }
    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    section = (b-a) / np;

    for (i=1; i<np; i++) {
        start = a + section * i;
        end = start + section;

        MPI_Send(&start, 1, MPI_INT, i, 1, MPI_COMM_WORLD);
        MPI_Send(&end, 1, MPI_INT, i, 1, MPI_COMM_WORLD);
    }
    for( int i=1; i<np; i++) {
      MPI_Recv(&part_area, 1, MPI_INT, i, 1, MPI_COMM_WORLD,
                                              &status);
      area = area + part_area;
    }
    cout << "area is " << area << endl;
// continued...
}
```

Q 4-20

```
if (rank != 0) {
    auto f = [](int x) { return 4/(1+x*x); };
    int delta = (b-a)/n;
    MPI_Recv (&start, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
    MPI_Recv(&end, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
    for (int i = start; i <= end; i += delta) {
        if ( i % 2 == 0)
            part_area = part_area + 2 * f(i);
        else
            part_area = part_area + 4 * f(i);
    }
    MPI_Send(&part_area, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
}
```

Ch6 : Synchronous Computations

Q 6-10 use finite difference eqt to solve one-dimensional problem using para-prog.



Assig 2

Q 6-10 continued...

```
float x_left = 0, x_right = 0;

if (rank == 1)      x_left = 10;
if (rank == 999)    x_right = 250;
Float x;
for (int i = 1; i < 1000; i++) {

    x = 0.5 * (x_left + x_right)

    isend (&x, rank-1)      if (i != 1)
    isend (&x, rank+1)      if (i != 1000-1)
     // blocking
    recv (&x, rank-1)       if (i != 1)
    recv (&x, rank+1)       if (i != 1000-1)

}
send(&x, 0)     // master proc can deal w it.
```