

COMP4300: Parallelisation via Data Partitioning

Chapter 5: Lin and Snyder
Chapter 4: Wilkinson and Allen

Alistair Rendell

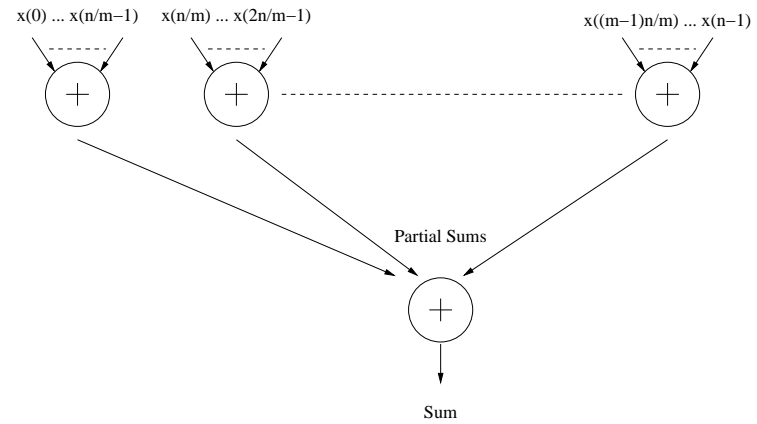
7.1 Parallelisation Strategies

- Replicated data approach
 - Each process has entire copy of data but does subset of computation
- Partition program data to different processes
 - Most common
 - Domain decomposition
 - Divide and Conquer
- Partitioning of program functionality
 - Much less common
 - Functional Decomposition
- Consider addition of numbers

$$sum = \sum_{i=0}^{n-1} x_i$$

7.2 D&C Example#1: Simple Summation of Vector

- Divide numbers into m equal parts



7.3 Master/Slave Send/Recv Approach

Master

```
s = n/m
for (i = 0, x = 0; i < m; i++, x = x + s)
    send(&numbers[x], s, i)
```

```
sum = 0;
for (i = 0; i < m; i++){
    recv(&part_sum, any_processor)
    sum = sum + part_sum;
}
```

Slave

```
recv(numbers, s, master)
part_sum = 0;
for (i = 0; i < s; i++)
    part_sum = part_sum + numbers[i];
send(&part_sum, master)
```

7.4 Using MPI_Scatter and MPI_Reduce

```
sendcount = n/m;
MPI_SCATTER(void *sendbuf, int sendcount, MPI_Datatype sendtype,
           void *recvbuf, int recvcount, MPI_Datatype recvtype,
           int root, MPI_Comm com)

for (i = 0; i < sendcount; i++)
    part_sum = part_sum +numbers[i];

MPI_REDUCE(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,
          MPI_Op MPI_SUM, int root, MPI_Comm comm)
}
```

- NOT Master/Slave
- Root sends data to all processes in MPI_Comm (including self)
- Note related MPI calls
 - MPI_SCATTERV scatters variable lengths
 - MPI_ALLREDUCE will return result to all processors

7.6 Domain Decomposition via Divide and Conquer

- Problems that can be recursively divided into smaller problems of the same type
- Recursive implementation of the summation problem

```
int add(int *s)
{
    if (numbers(s) <= 2) return (s[0]+s[1]);
    else{
        divide(s, s1, s2);
        part_sum1 = add(s1);
        part_sum2 = add(s2);
        return (part_sum1 + part_sum2);
    }
}
```

7.5 Analysis

Sequential

- $n - 1$ additions thus $O(n)$

Parallel

- Communications#1:

$$t_{comm1} = m(t_{stup} + (n/m)t_{data})$$

- Computation#1:

$$t_{comp1} = n/m - 1$$

- Communication#2:

$$t_{comm2} = m(t_{stup} + t_{data})$$

- Computation#2:

$$t_{comp2} = m - 1$$

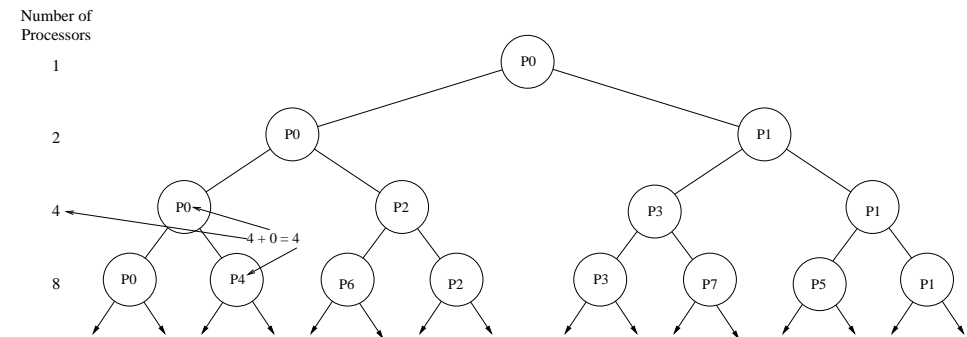
- Overall

$$t_p = 2mt_{stup} + (n + m)t_{data} + m + n/m - 2 = O(n + m)$$

- Worse than sequential code!!

7.7 Binary Tree

- Divide and conquer with binary partitioning
- Note number of working processors decreases going up the tree



7.8 Simple Binary Tree Code

```

\* Binary Tree broadcast
a) 0->1
b) 0->2, 1->3
c) 0->4, 1->5, 2->6, 3->7
d) 0->8, 1->9, 2->10, 3->11, 4->12, 5->13, 6->14, 7->15
*\
lo = 1
while (lo < nproc) {
  if (me < lo) {
    id = me + lo;
    if (id < nproc) send(type, buf, lenbuf, &id);
  }
  else if (me < 2*lo) {
    id = lo;
    recv(type, buf, lenbuf, &lenmes, &id);
  }
  lo *= 2;
}

```

7.9 Analysis

- Assume n is a power of 2 and ignoring t_{stup}
- Communication#1: Division

$$t_{comm1} = \frac{n}{2}t_{data} + \frac{n}{4}t_{data} + \frac{n}{8}t_{data} + \dots + \frac{n}{p}t_{data} = \frac{n(p-1)}{p}t_{data}$$

- Communication#2: Combining

$$t_{comm2} = t_{data} \log p$$

- Computation:

$$t_{comp} = \frac{n}{p} + \log p$$

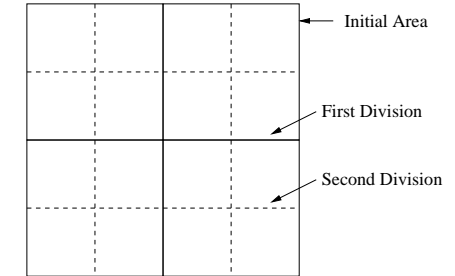
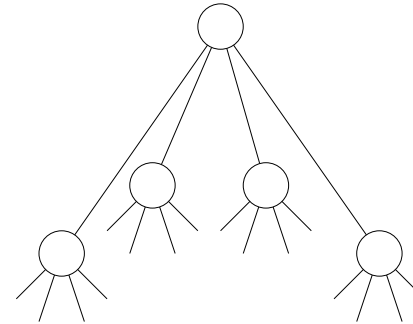
- Total:

$$t_p = \frac{n(p-1)}{p}t_{data} + t_{data} \log p + \frac{n}{p} + \log p$$

- Slightly better than before - as $p \rightarrow n$ cost $\rightarrow O(n)$

7.10 Higher Order Trees

- Possible to divide data into higher order trees, eg quad tree



7.11 The Reduce and Scan Abstractions

- Summation of a vector already partitioned between processes is an example of the reduce and scan abstraction
- Reduce:** Combines a set of values to produce a single value
- Scan:** performs a sequential operation in parts and carries along the intermediate results
- Reduce usually involves mapping a binary (or higher level) tree communication pattern between processes
- Examples (to discuss) include
 - Finding second smallest array element
 - Computing a k-way histogram
 - Length of longest run of 1s
 - Index of first occurrence of x
- See Lin and Snyder for further details

7.12 D&C Example#2: Bucket Sort

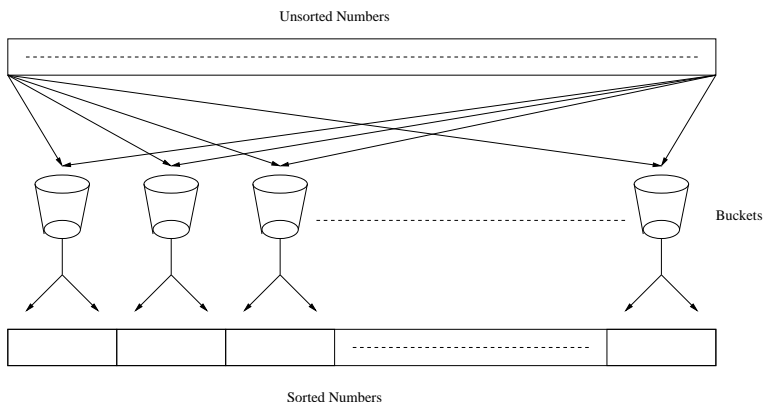
- Divide number range (a) into m equal regions

$$\left(0 \rightarrow \frac{a}{m} - 1\right), \left(\frac{a}{m} \rightarrow 2\frac{a}{m} - 1\right), \left(2\frac{a}{m} \rightarrow 3\frac{a}{m} - 1\right), \dots$$

- Assign one bucket to each region
- Stage 1: numbers are placed into appropriate buckets
- Stage 2: each bucket is sorted using a traditional sorting algorithm
- Works best if numbers are evenly distributed over the range a
- Sequential time

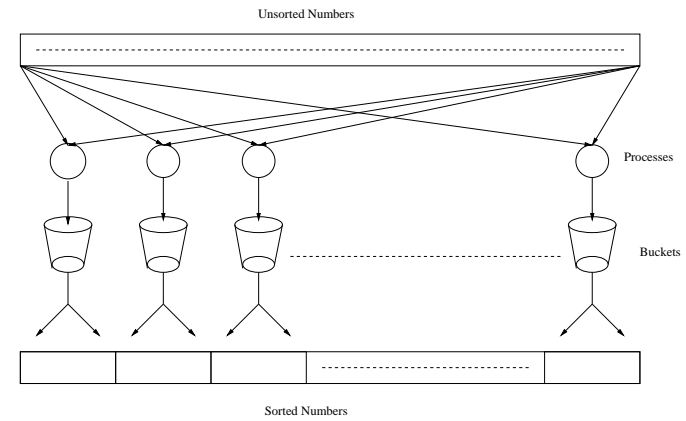
$$t_s = n + m((n/m) \log(n/m)) = n + n \log(n/m) = O(n \log(n/m))$$

7.13 Sequential Bucket Sort

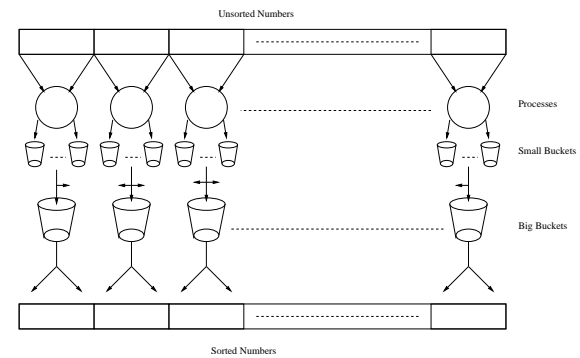


7.14 Parallel Bucket Sort#1

- Assign one bucket to each process



7.15 Parallel Bucket Sort#2



- Assign p small buckets to each process
- Note possible use of MPI_ALLTOALL

```
MPI_Alltoall(void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, int recvcount, MPI_Datatype recvttype,
             MPI_Comm comm)
```

7.16 Analysis

- Initial partitioning and distribution

$$t_{comp1} = n$$

$$t_{comm1} = t_{stup} + t_{data}n$$

- Sort into small buckets

$$t_{comp2} = n/p$$

- Send to large buckets: (overlapping communications)

$$t_{comm3} = (p-1)(t_{stup} + (n/p^2)t_{data})$$

- Sort of large buckets

$$t_{comp4} = (n/p) \log(n/p)$$

- Total

$$t_p = t_{stup} + t_{data}n + n/p + (p-1)(t_{stup} + (n/p^2)t_{data}) + (n/p) \log(n/p)$$

- At best $O(n)$
- What would be the worse case scenario?

7.18 Static Distribution: SPMD Model

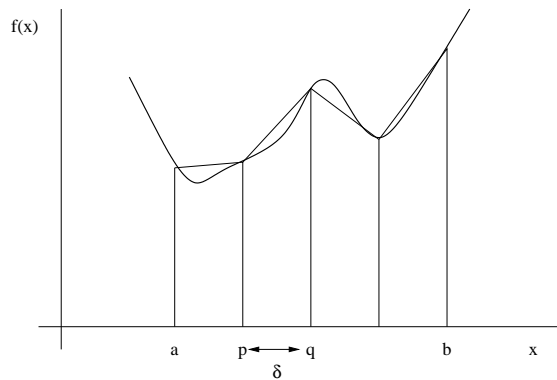
```

if (i == master){
    printf(" Enter number of regions\n");
    scanf("%d",&n);
}
broadcast(&n,process_group)
region = (b-a)/p;
start = a + region*i;
end = start + region
d = (b-a)/n
for (x = start; x < end; x = x + d)
    area = area + f(x) + f(x+d);
    area = 0.5 * area *d;
reduce_add(&integral, & area, processor_group);
    
```

7.17 D&C Example#3: Integration

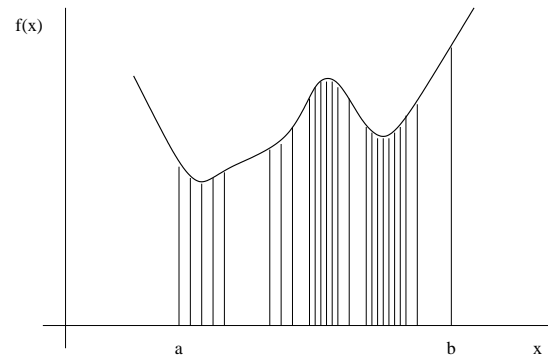
- Consider evaluation of integral using trapazoidal rule

$$I = \int_a^b f(x)dx$$



7.19 Adaptive Quadrature

- Not all areas require the same number of points
- When to terminate division into smaller area is an issue
- Parallel code will have uneven work load



7.20 D&C Example#4: N-Body Problems

- Summing longrange pairwise interactions, eg gravitation

$$F = \frac{Gm_a m_b}{r^2}$$

where G is the gravitational constant, m_a and m_b are the mass of two bodies, and r is the distance between them

- In Cartesian space

$$F_x = \frac{Gm_a m_b}{r^2} \left(\frac{x_b - x_a}{r} \right)$$

$$F_y = \frac{Gm_a m_b}{r^2} \left(\frac{y_b - y_a}{r} \right)$$

$$F_z = \frac{Gm_a m_b}{r^2} \left(\frac{z_b - z_a}{r} \right)$$

- What is the total force on the sun due to all other stars in the milky way?
- Given the force on each star we can calculate their motions
- Molecular dynamics is very similar but long range forces are electrostatic

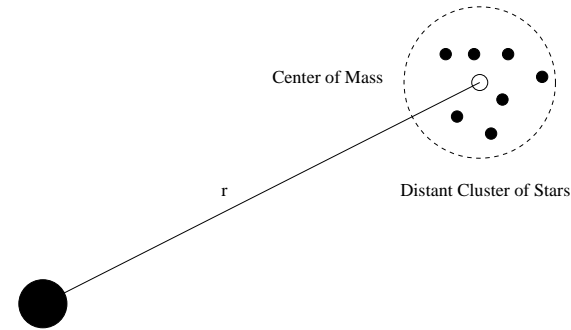
7.21 Simple Sequential Force Code

```
for (i = 0; i < nstars; i++)
  for (j = 0; j < nstars; j++){
    if (i != j){
      rij2 = (x[i]-x[j])*(x[i]-x[j])
        + (y[i]-y[j])*(y[i]-y[j])
        + (z[i]-z[j])*(z[i]-z[j]);
      Fx[i] = Fx[i] + G*m[i]*m[j]/rij2 * (x[i]-x[j]) / sqrt(rij2)
      Fy[i] = Fy[i] + G*m[i]*m[j]/rij2 * (y[i]-y[j]) / sqrt(rij2)
      Fz[i] = Fz[i] + G*m[i]*m[j]/rij2 * (z[i]-z[j]) / sqrt(rij2)
    }
  }
```

- Aside: How could you improve this sequential code?
- $O(n^2)$ - this will get very expensive for large N
- Is there a better way?

7.22 Clustering

- Idea: the interaction with several bodies that are cluster together but are located at large r for another body can be replaced by the interaction with the center of mass of the cluster



7.23 Barnes-Hut Algorithm

- Start with whole space in one cube
 - Divide the cube into 8 subcubes
 - Delete subcubes if they have no particles in them
 - Subcubes with more than 1 particle are divided into 8 again
 - Continue until each cube has only one particle (or none)
- Process creates an *octree*
- Total mass and centre of mass of each subcube stored at each node
- Force evaluated by starting at each root and traversing the tree, BUT stopping at a node if the clustering algorithm can be used
- Scaling $O(n \log n)$
- Load balancing likely to be an issue for parallel code

7.24 Barnes-Hut Algorithm: 2D Illustration

