Designing Parallel Program

Tutorial

Lec 2

Agenda

- Designing parallel program
- Example KMeans clustering
- Parallel Efficiency

Agenda

- Designing parallel program
- Example KMeans clustering
- Parallel Efficiency

Parallel Program Design

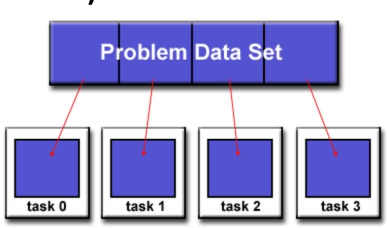
1. Understanding the Program
2. Partitioning strategies
3. Load Balance
4. Data Dependencies
5. Synchronization

1- Understanding the Program

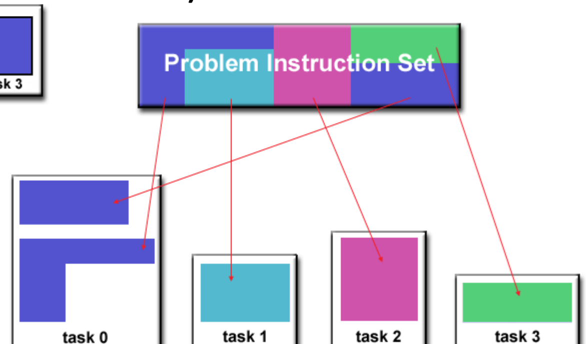
- Identify the program's ***hotspots***:
  - Identify the portion of your code that consumes most of the time
    - With help of Profilers
- Identify ***bottlenecks*** in the program:
  - Identify the areas in your code that slows down your program i.e. I/O operations or serial regions
- Investigate other algorithms if possible.
- Take advantage of optimized third party parallel software i.e. BLAS, MKL etc

2- Partitioning strategies

1) Data Partition

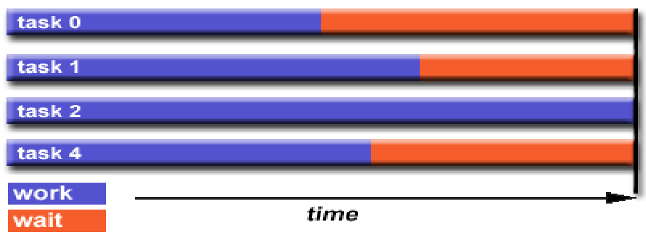


2) Function Partition



3- Load Balance

- Are all processes getting equal amount of time.
- Uneven work distribution can cause some parallel processes to wait for others to complete.



4- Data Dependencies

- If the order of execution of program statements affects the results of the program calculation than there exist dependency.
- In particular Data decencies happen when same memory location is accessed multiple time and order of access is curtail.

```
for(int i =1; i < array.length-1; i++){
  array[i] = array[i] +  array[i-1];
}
array[1] = array[1] +  array[0];
array[2] = array[2] +  array[1];
array[3] = array[3] +  array[2];
```

5- Synchronization

- Barrier
  - Synchronize all workers at a point in program execution
- Lock / semaphore
  - Avoid data race or deadlocks
- Synchronous communication operations
  - Synchronization caused by process communication with eachother

Agenda

- Designing parallel program
- Example KMeans clustering
- Parallel Efficiency

Kmeans Clustering Algorithm

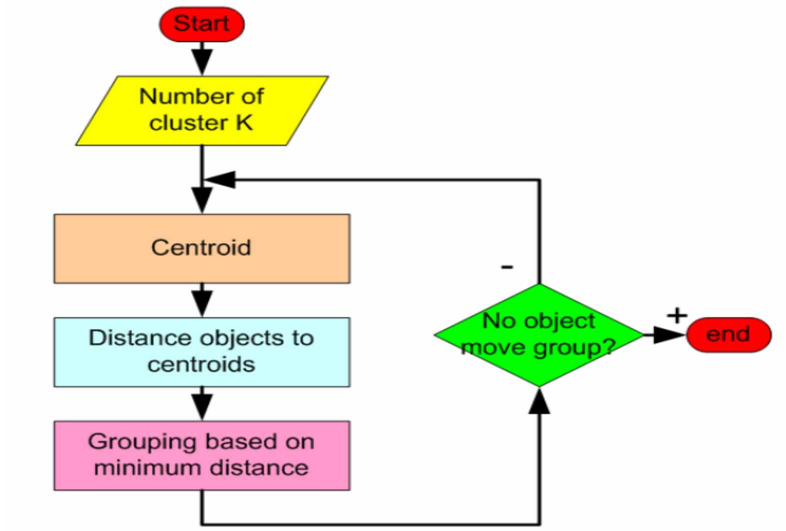


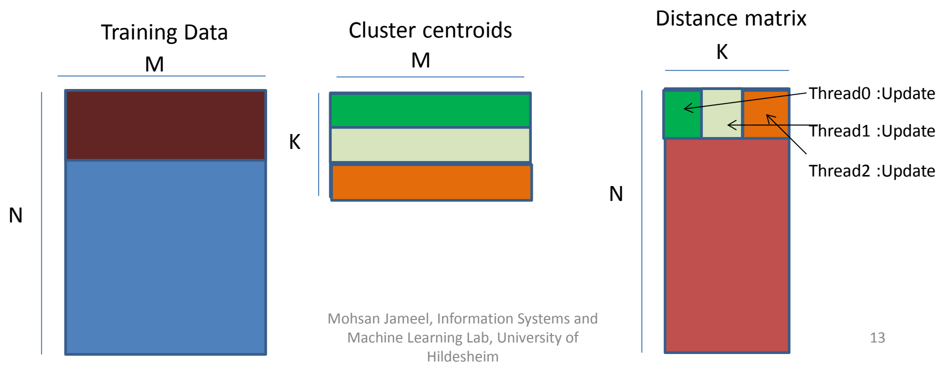
Figure ref: http://croce.ggf.br/dados/K%20mean%20Clustering1.pdf

Kmeans Clustering

- initCenters()
  - Randomly initialize centroids of clusters.
- computeDistance()
  - Compute distance of each training example to all the centroids
  - Assign each training example to one of cluster based on minimum distance.
- computeCenter()
  - Update the centroids of clusters based on new assignments.

computeDistance()

- In computeDistance() only Distance matrix is updated.
- Each entry of Distance matrix can be updated in parallel since there is no data dependency between their updates.
- Training Data and Cluster centroids only accessed as read only.



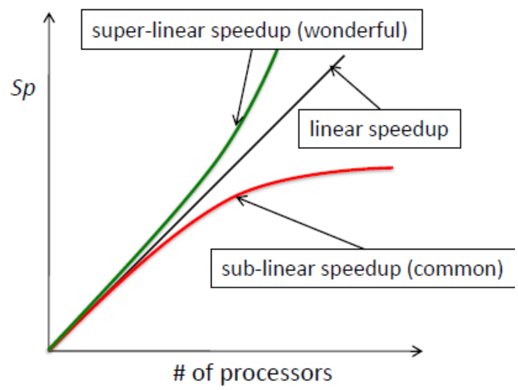
Agenda

- Designing parallel program
- Example KMeans clustering
- Parallel Efficiency

Parallel Speedup & Efficiency

- Speedup $S_p = \frac{T_s}{T_p}$
  - P = # processes
  - Ts = Best serial execution time
  - Tp = execution time on P processes
  - Sp = Speedup on P processes
- Parallel Efficiency Ep

$$E_p = \frac{S_p}{p} = \frac{T_s}{pT_p}$$



Amdahl's Law

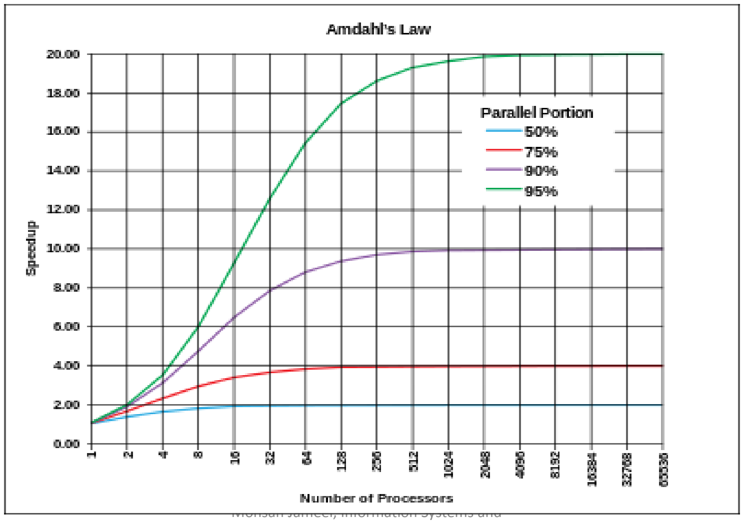
- Amdahl's law give theoretical limit of parallel program for **fixed workload**
- All parallel programs contain:
  - parallel sections
  - serial sections
- Serial sections limit the parallel effectiveness
- Amdahl's Law states this formally

$$S_{\text{latency}}(s) = \frac{1}{1 - p + \frac{p}{s}}$$

- Example
  - Let p =0.30 30% (portion of task to be performed in parallel
  - Let s = 2 times speedup of parallel portion

$$S_{\text{latency}} = \frac{1}{1 - p + \frac{p}{s}} = \frac{1}{1 - 0.3 + \frac{0.3}{2}} = 1.18.$$

Amdahl's Law



Gustafson's Law

- Effect of multiple processors on run time of a problem with a ***fixed amount of parallel work per processor***

$$S_{\text{latency}}(s) = 1 - p + sp,$$

- Limitation: Some problem does not have significantly large dataset

References

- http://croce.ggf.br/dados/K%20mean%20Clustering1.pdf
- https://en.wikipedia.org/wiki/Amdahl%27s_law
- https://computing.llnl.gov/tutorials/parallel_comp/
- https://en.wikipedia.org/wiki/Gustafson%27s_law