Parallel Programming (SE 295)
Assignment 03

Satish Kumar
M.Tech. SERC
SR No. -11052

Programming Strategies

A wider look of the parallel nested dissection strategy along with parallel KL is being shown in the scanned image on next page. From Implementation point of view, This assignment involves various sub-tasks.

Harwell-Boeing matrix and Data Structure handling

- Since Harwell-Boeing sparse matrix data obtained from directory is arranged in **CSR** Formate, For each matrix, Row Pointer array and Collumn index array can be read directly from the file, and used for further processing. I have a MATLAB program to do this task.
- I have worked with some set of large symmetric data and I have used **(n=28924 , nnz=1036208)** matrix `http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc5/bcsstk30.html` for reporting the results.
- Since all the graph work has been done with the reference to CSR form of matrix, When i need to **reorder** the nodes of corresponding graph, I have used a **mapping array** P. Here, P[i] = j means, node i has been reordered to j.

KL Algorithm

A routine to perform parallel KL algorithm has been implemented. Here,

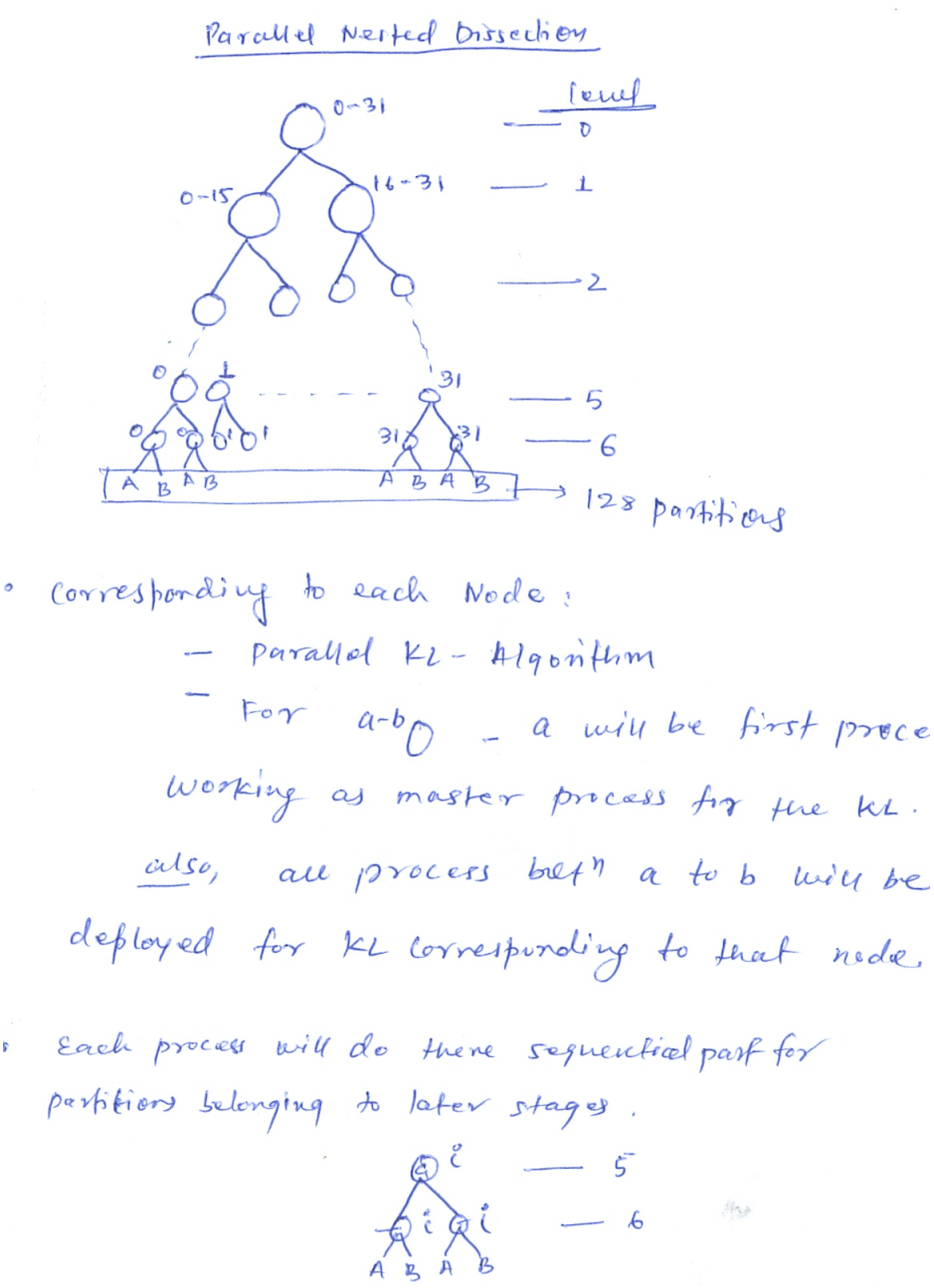
Inputs:
1. A graph(G) in CSR Form (Only row pointer and collumn index array)
2. No. of Processes for parallel computations
3. Process ID for First Process in the set of processes

Outputs:
1. Set of nodes A, B and v, where $\{G\} = \{A\}U\{B\}U\{v\}$. KL algorithm make sure that A and B are disjoint set of nodes and separator sets of node v are minimum.
2. A permutation array which contains mapping for reordering of nodes with respect to original nodes.

Strategies:

- Steps of computations are same as discussed in the class (please refer to the steps mentioned in the slide, sparseLA.ppt, Page No. 47).
- For parallel version, only two steps has been consider for parallel computations: 1. Computing cost of D(n) - $O(N^2)$ 2. Finding maximum gain pair (a,b) - $O(N^2)$. Other steps has $O(N)$ complexity.
- Computing D(n) doesn't involve any communications since entire graph data is already available to each processes. Hence no communication is required.
- For maximum gain pair (a,b), We need to broadcast D(n) data. Once each process has entire D(n) data, Each process will find maximum gain pair (a',b) for $\|a'\| = \frac{n}{nproc}$. Finally MPI_AllGather to the first process to find ultimate maximum pair (a,b).
- For selecting separator nodes v, from new A and B, I have considered all the pair nodes (a,b) having an edge (none zero value). Then selected one node of the pair alternatively belonging to set A and B.

continued on next page...



Parallel Nested Dissection

- Since We have to reorder the graph with 128 partition, Idea of the implementation is as shown in the scanned image above.
- Different Parallel KL computations along with parallel computations for each partitions has no. of advantages due to full utilization of processes. Load is always balanced.
- Each process contains entire permutation array. Each process also have another array of size n, lets say Pb, which keeps labels specifying whether a node belongs to computations for that process or not.

continued on next page...

Here, Pb[i] = -1 means i^{th} node belongs to there local A, +1 means local B, 0 means i^{th} node doesn't belong to that process.

- Each Pb array returns with 4 type of label: local A, local B, local Separator v and Nodes that doesn't belong to that partition.
- Here, Permutation matrix - which contains reordering information, is getting updated by each processes for their partitions. Hence, In each level, Permutation matrix must be communicated to a master process(the first process) so that it can further gets split into 2 partitions.
- Due to limitation in no. of processes, after level 5 (32 processes) each process must do separate sets of sequential KL parallely with more than 1 KL on same level (near leaf nodes). Each process has to do sequential KL for their portion of sub-tree (3 partitions).

Minimum Degree Reordering

- Please refer to the `https://www.cs.ubc.ca/~sfingram/cs517_final.pdf` for steps involved in Minimum Degree Reordering. Paper suggest a concept of "enode" to avoid deletion of node and constructing new edges. Hence reducing complexity.
- Updating degree with handling enodes is the main task here.
- For tie breaking, I have selected the first node index-wise, from the set of nodes of minimum degree.

Finding No. of fills

- The idea of enode is also helpful in finding no. of fills with reduced complexity.
- Here, We have to delete nodes (virtually enode) from 0 to n-1 (unlike minimum degree node), And for each deletion, We have to see commutative increase in degree of reachable nodes (Neighbors Node). That will give us no. of fills.
- Again, to deal with various reordering, I have used permutation array.

Results:

Though my program is working for sequential version implemented, I am getting **errors** in parallel version of nested dissection.

Execution Time

Parallel Nested Dissection: —
Sequential Minimum Degree Reordering: 33.25 seconds
SppedUp: —

No. of fills

Original Matrix: 1218292
After Nested Dissection: 42136
After Min-Degree Reordering: 78331

continued on next page...

Observations

- Nestedd dissection reordering process is of $O(n^2)$ whereas minimum degree reordering in worst case can go up $O(n^2 * nnz)$.
- Idea of full utilization of process gives better load balance for parallel nested dissection with parallel KL.
- Minimum degree reordering gives lesser fill-ins compared to nested dissection with few partitions. But as we increse no. of partitions, Nested dissection gives lsser and lesser no. of fills. It can be noted that Nested dissection is an improved version of minimum degree reordering with better tie-breaker.
- There are better tie-breaking strategies available for minimum degree reordering like MMU, AMU. It has been observed that Efficiency of algorithm is very much dependent on tie-breaking strategy.

END.