Distributed Systems

Monsoon 2024

Lecture 25

International Institute of Information Technology

Hyderabad, India

Some Terminology

- We have seen some classification of distributed algorithms as
  - Synchronous vs. Asynchronous
- More such categories can be seen in the literature and in applications.
- Often these categories may be non-overlapping with each other.

Some Terminology

- Uniform algorithms: These are distributed algorithms, in particular, graph algorithms, where the algorithm does not use any knowledge of the graph topology
  - Such as the number of nodes, diameter, number of links, degree, etc.
  - Uniform algorithms scale naturally since any changes in the input need not require any changes to the algorithm or in any step of its execution.
    - For instance, if a node has to choose a particular action with a probability based on the degree of the graph, then any changes to the graph may change the degree.

Some Terminology

- Symmetric vs Asymmetric Algorithms
  - A distributed algorithm is said to be symmetric if all nodes perform similar roles.
  - Otherwise the algorithm is asymmetric.
  - Example: Luby's algorithm is symmetric, whereas the source initiated BFS algorithm is asymmetric.

Some Terminology

- Anonymous algorithms: Such distributed algorithms do not need any node identifiers in their execution.
- Usually tough and many instances impossible to design algorithms in the anonymous setting.
  - Most algorithms use node ids to break ties!
  - Several impossibility results are known in this strong setting.

Models for Distributed Algorithms

- LOCAL
- CONGEST
- CONGESTED-CLIQUE
- Massively Parallel Computing Model

LOCAL Model

- Each round consists of local computation and message send/receive events.
- Local computation, polynomial or more, is completely ignored in understanding the complexity.
- The volume/size of messages are not taken into account.
- Any node can send a message of size polynomial in the input size in one round to each neighbor.
- The parameter of efficiency of an algorithm is the number of rounds required to complete.

LOCAL Model

- For most graph problems, a trivial upper bound therefore would be the diameter of the graph.
- Each node sends its list of neighbors to a special designated nodes, say node number 1.
- Node number 1 receives the neighbor lists from each node. It can then assemble the graph.
- Node number 1 can then compute the required answer and if needed, send the answer to all other nodes.
- However, it is possible to do better in many cases.

LOCAL Model

- For MIS for instance, one can show an upper bound of O(sqrt{log n} log Δ) rounds in the randomized rounds.
  - Algorithms are difficult to analyze.
  - A trick used in such fast algorithms is to run a usual slow algorithm for a few rounds.
  - At the end of these few rounds, carefully analyze the progress made by the algorithm.
  - Obtain bounds on the properties of the left over graph.
  - Use the fact that message sizes can be large to gather the graph at a single node, solve locally, and distributed the answer.
- Lower bounds are much more tricky to prove.
  - Lower bounds for a few problems are known.

CONGEST Model

- Here too, the network representing the distributed system is abstracted as a graph.
- The nodes of the network correspond to the nodes in the graph, and the communication links are the edges of the graph.
- In a departure from the LOCAL model, here it is assumed that the bandwidth of the links is limited.
- So, messages are limited in size and bigger messages require more rounds.
- The bandwidth of each link is usually limited to O(log n) where n is the number of nodes in G.

CONGEST Model

- In a departure from the LOCAL model, here it is assumed that the bandwidth of the links is limited.
- So, messages are limited in size and bigger messages require more rounds.
- The bandwidth of each link is usually limited to O(log n) where n is the number of nodes in G.
- O(log n) is the size required to send the id of a node of G with n nodes. Hence, the limit is not artificial.

CONGEST Model

- Non-trivial lower bounds are not known for many problems.
- Algorithm design also more difficult than the LOCAL model.

- For example, Luby‘s MIS algorithm can be made to run in the CONGEST model.

CONGESTED-CLIQUE

- This model is significantly different from the LOCAL and the CONGEST models.
- The model uses a clique graph of n nodes with communication links connecting each pair of nodes.
- This model is one of the so called All-to-All Communication models.
- The input graph is a spanning subgraph of this graph.
  - In other words, the input graph too has n nodes but possibly fewer links.
- However, all communication links can be used to solve the problem of interest.

CONGESTED-CLIQUE

- The input graph is a spanning subgraph of this graph.
- However, all communication links can be used to solve the problem of interest.
- Each link however has a limited bandwidth of O(log n) bits.
- The extra communication links can be used to quickly send information but the size of the message in each round is limited.

Some Open Issues

- The CONGESTED-CLIQUE model is a bit too relaxed and artificial.
- Usually, expanding the communication network as a clique is difficult. Not scalable in practice.
- However, one can create what are called as expanders easily
- An expander graph has the property that any two n/2 sized subsets of the graph have a good connectivity between them
  - Such graphs with low degree, as low as O(1), also exist!

A Model for Cloud Computing

- It is important to connect distributed computing models to practical models like the cloud.
- One step in that direction is the Massively Parallel Computing model, MPC model for short.
- The MPC model has a number of pairwise interconnected machines.
  - Similar to the All-to-All communication model
- However, we assume that no single machine can store the entire input by itself.
  - Space is not enough!
- We still measure the number of rounds of communication needed to solve a problem.

The MPC Model

- Let the input have a size of N.
- Let each machine have a space of S. We assume that N >> S.
- Each machine can in each round send and receive up to S units of data via its links.
- The input is distributed across the machines in equal proportion with each of the O(N/S) machines holding O(S) data.
- When the input is a graph, notice that N = O(m+n).
- There are three variants studied in this context based on the relation between n and S.

The MPC Model

- The super-linear regime: Here, $S = \Omega(n^{1+\varepsilon})$.
- We still assume that $m >> S$.
- So, each machine has enough space to store information about all vertices and more than that.
- How is this additional space helpful?
- Let us study with the MST problem again.
- Partition the edges of the graph into $E_1, E_2, \ldots, E_r$ such that $|E_i| = n^{1+\varepsilon}$.
- Compute $T_i = MST(E_i)$.
- Recursively compute and return the MST of $U_i T_i$.

The MPC Model

- How many recursive calls are needed?
- Assume $m = n^{1+c}$.
- So, initially, there are $m/n^{1+\varepsilon} = n^{c-\varepsilon}$ sets of edges.
- After computing T_i with at most n edges each, the number of edges in $U_i T_i$ is at most $nxn^{c-\varepsilon} = n^{1+c-\varepsilon}$.
- So, each recursive call reduces the number of edges from n^{1+c} to $n^{1+c-\varepsilon}$.
- In c/ε such recursive calls, the number of edges falls to a quantity where a single machine can store all the edges.
- Algorithms that run in $O(1/\varepsilon)$ rounds are known for several other graph problems too in this model.