DSC-204A: Scalable Data Systems, Winter 2024

11: Collective Communiation - I

Lecturer: Hao Zhang *Scribe: Anuj Goenka, Ruben Chatterjee, Abhishek Jadhav*

1 Recap: Networking

Some True/False questions were discussed to have a quick recap of the last lecture:

1. **Protocol specifies the implementation.**
   Answer: **False**, Protocol just specifies the instructions and the implementation is specified by the processes.

2. **Congestion control takes care of the sender not the overflowing the receiver.**
   Answer: **False**, Congestion control takes care of routers. The flow control takes care of the sender not overflowing the receiver.

3. **A random access protocol is efficient at low utilization.**
   Answer: **True**, This is true because there are fewer collisions.

4. **At the data link layer, hosts are identified by IP addresses.**
   Answer: **False**, IP address is defined in the Network layer.

5. **The physical layer is concerned with sending and receiving bits.**
   Answer: **True**

6. **Layering improves application performance.**
   Answer: **False**: It doesn't improve application performance because of abstraction, we need to go deeper and deeper for implementation details.

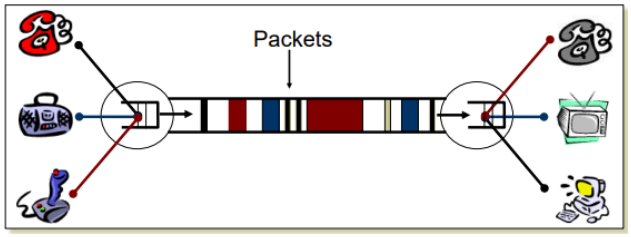
7. **Routers forward a packet based on its destination address.**
   Answer: **True**

8. **"Best Effort" packet delivery ensures that packets are delivered in order.**
   Answer: **False**, Application and Transport Layer ensure that delivery of packets are done in order.

9. **Port numbers belong to network layer.**
   Answer: **False**, Port numbers belong to Transport layer.

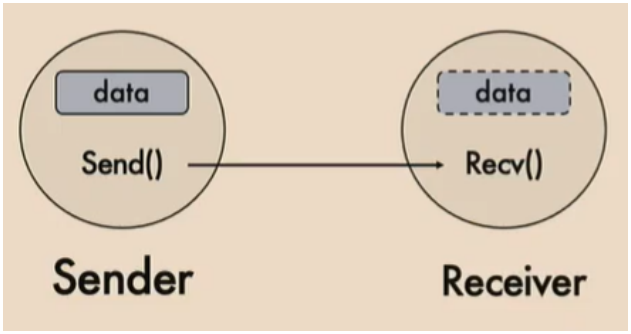
2 Communication

In the figure, we can see the type of communication is peer-to-peer communication. For example, when we call someone, there's only one sender and one receiver in the communication channel.



2.1 Point-to-point communication

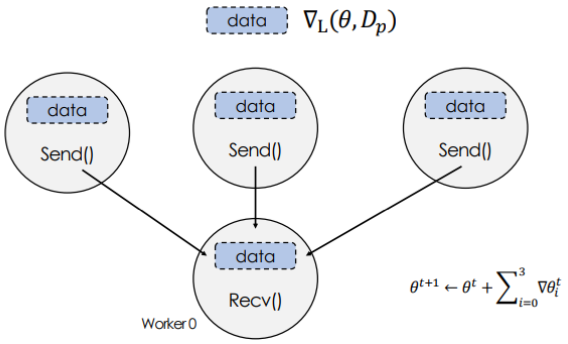
Also known as peer-to-peer communication. There are only two participants: the Sender and the Receiver.
How is it implemented?
We implement two functions: send() and recv(). These functions first establish a TCP connection which ensures that the data is eventually received by the receiver. Then the application starts sending data and the data goes down through 5 layers, through the network, and arrives at the receiver.



How is it implemented in Ray?
Implement two functions send and receive. Now we can use Ray object store. On the sender side, we will put the data into the shared memory, using API 'ray.put()' and on the receiver's side, we will use the function 'ray.get()' to get the reference to the data.
Example of Gradient Equation
The gradient equation is given by:

$$\theta^t = \theta^{t-1} + \epsilon.\nabla_L(\theta^{t-1}, D^t)$$

Now in case of big data. The data is distributed across many CPUS and GPUs(An example is shown in the figure) and each machine computes the gradient of the partition of the data provided. This concept is also known as data parallelism.



Now the equation becomes:

$$\theta^t = \theta^{t-1} + \epsilon \sum_{p=1}^{p} \nabla_L(\theta^{t-1}, D_p^t)$$

We can't directly perform the sum. So we can consider this a network with 3 senders and 1 receiver. The 3 sender will communicate their data to the receiver and the receiver will add all the data. This is a type of collective communication and is known as 'Reduce'. When we analyze the performance of this method, We get Message over Network as 3*N.

For synchronization purposes, using Reduce, for all nodes to have the collective data (the sum of the data). We have to make every node a worker(receiver) node simultaneously using a loop. Hence, we get Message size over the Network as 4 times 3*N i.e. 12N. This is not the ideal case. Instead, we can use 'AllReduce' collective communication. What happens is that the senders send their respective data to the receiver and the receiver performs the sum of the data and broadcasts it over the network. When we analyze the performance of this method, we get the Message over the Network as 6*N. A better case.

3 Why we need Collective Communication

1. For Programming Convenience-

Communication between a group of participants is carried out by composing communication patterns using different scenarios. This requires extensive coding and logic to be implemented. 20 years ago, this was very difficult to achieve in low level languages like C, where the code became too long. Introduction of Ray gives a more well defined structure to this reducing task and hence ensured programming convenience.

2. For Performance-

The send and receive pattern may not always be the most optimal way to carry out collective communication. We can define a set of primitives and enforce people to carry out their communication in accordance with these primitives, which can be optimised over time, improving performance.

3. Machine Learning Systems love collective communication-

Machine Learning and collective communication go hand in hand. Even when 50 years ago, there was no distributed computing, networking people realised the patterns were same and comparing this in today's time, it is very similar to training large models.

4 Model of Parallel Computation

1. a node can send directly to any other node (maybe not true)
2. a node can simultaneously receive and send
3. cost of communication (sending a message of length n between any two nodes)

$$Cost = \alpha + n\beta$$

4. If a message encounters a link that simultaneously accommodates M messages, the cost becomes

$$Cost = \alpha + Mn\beta$$

5 Prominent Collective Communication Methods

Broadcast: In the broadcast operation, one node (the root) sends a message to all other nodes in the network.

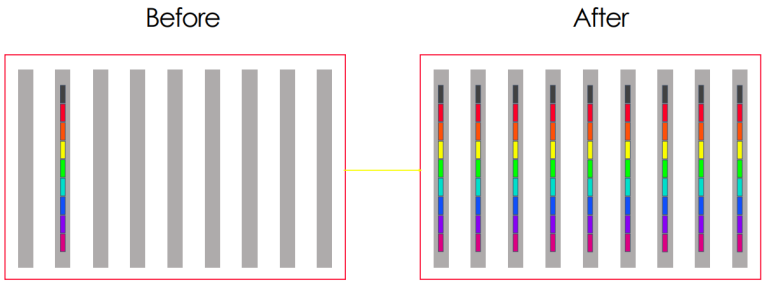


Figure 1: Broadcast

Reduce: In the reduce operation, data from all nodes are aggregated into a single result using a specific binary operation. It is the conjugate of broadcast.

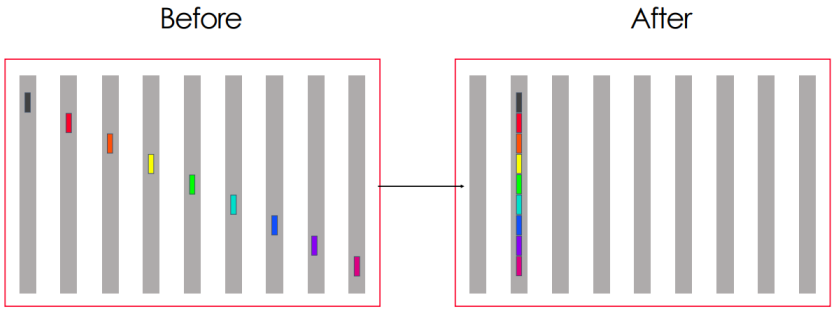


Figure 2: Reduce

Scatter: Scatter operation involves one node distributing data (n) to all other nodes in the network, with each node receiving a subset of the data (1/n). The data to be scattered is typically divided into equal-sized chunks and distributed among the nodes.

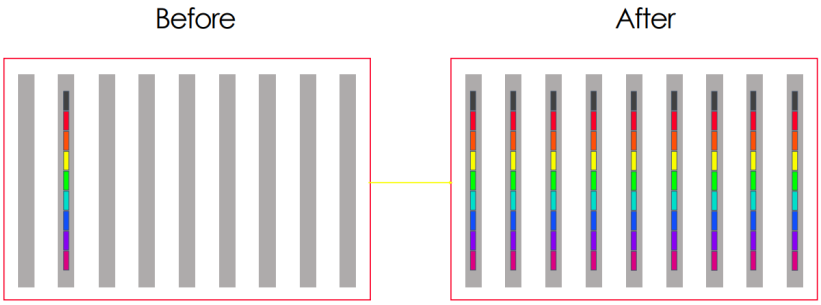


Figure 3: Scatter

Gather: Gather operation gathers data(1/n) from all nodes onto a single root node. Each node sends its local data to the root node, which collects and combines the data into a single result(n). It is the conjugate of scatter.

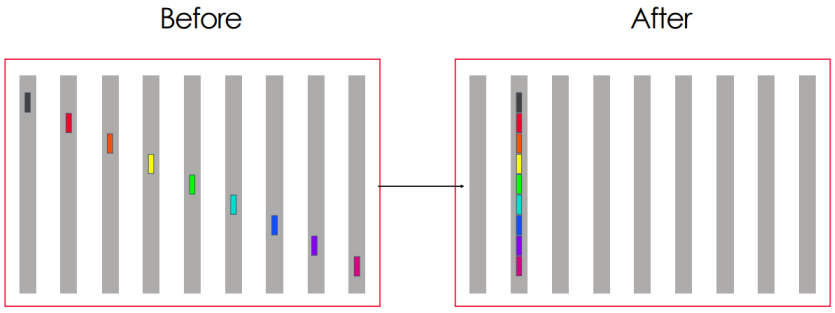


Figure 4: Gather

Allgather: Allgather operation is similar to gather where every node has a part of the message (1/n), and after applying Allgather, every node ends up with a copy of the entire message(n).

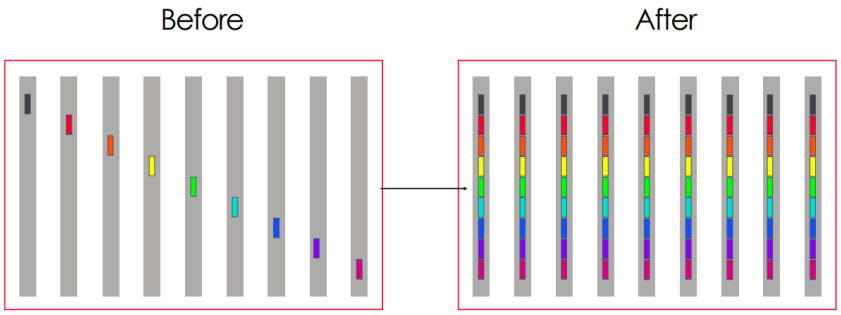


Figure 5: Allgather

Reduce-Scatter: Reduce-Scatter combines elements of both reduce and scatter operations. It first performs a reduction operation on the data across all nodes, producing partial results on each node. Then, it scatters these partial results among the nodes, ensuring that each node receives a portion of the final reduced data. It is the conjugate of Allgather.

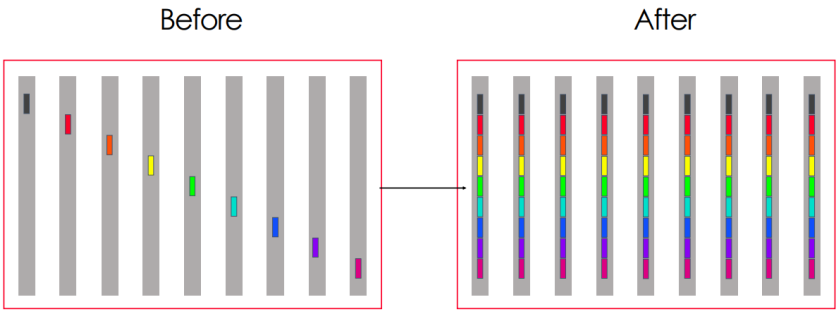


Figure 6: Reduce-Scatter

Allreduce: Allreduce combines elements of both reduce and broadcast operations. It performs a reduction operation on the data across all nodes, similar to the reduce operation. However, unlike reduce, the result of the reduction is then broadcasted to all nodes, ensuring that each node receives the final reduced result.

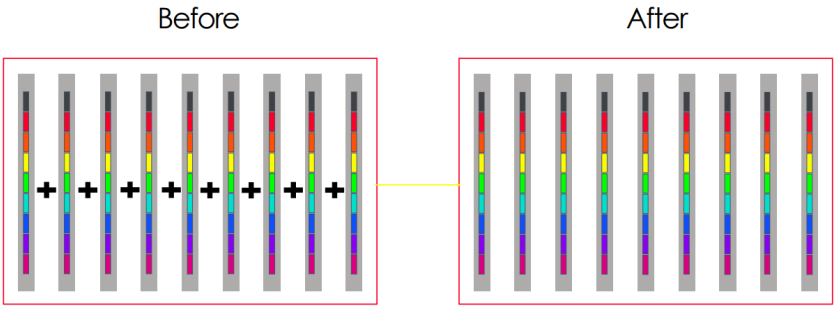


Figure 7: AllReduce

6 Algorithms for collective communication

There are two types of spaces where algorithms are developed for collective communication:

- Small Message Space: In this space, the size of the message itself is not huge, so preference is given to the latency of communication. The Minimum Spanning Tree algorithm falls into this category.
- Large Message Space: Here, the message size is large, so emphasis is given to bandwidth for faster communication. The Ring algorithm falls into this category.

Two prominent libraries that implement these algorithms are MPI and NCCL.

7 General principles of Minimum Spanning Tree

Minimum Spanning Tree follows the divide and conquer rule. It is advantageous for achieving optimum latency, as the best we can achieve is $\log n$ for sending messages to all n nodes.
Performance Analysis:

- Red arrows represent startup communication cost α (latency).
- Green arrows indicate transit cost β (bandwidth).

Broadcast: MST can be used for broadcasting in a recursive divide and conquer manner. For n number of nodes, we will need $2^t = n$ runs, where t is the number of runs. For example, for 8 workers, only 3 runs are needed, but for odd number of workers, it needs an additional run.

The cost of broadcast is given by:

$$\text{Cost} = (\log(p))(\alpha + n\beta) \tag{1}$$

Reduce: For reduce operation, we first divide and conquer, then sum the messages with a latency of α. This method is advantageous as it utilizes unused bandwidth in comparison to the P2P. The cost of reduce operation is:

$$\text{Cost} = \log(p)(\alpha + n\beta + n\gamma) \tag{2}$$

where γ is the cost of the reduce operation.

Scatter: Similar to reduce, scatter utilizes MST approach but sends only half of the message to the other side after each divide and conquer. The cost is given by:

$$\text{Cost} = \log(p)\left(\alpha + \frac{p-1}{p}n\beta\right) \tag{3}$$

Gather: Gather operation is similar to reduce, but after divide and conquer, a merge operation is performed. The cost is the same as scatter.

Allgather, Reduce-Scatter, and Allreduce: Allgather is the addition of the cost of gather and broadcast. Similarly, reduce-scatter is the addition of the cost of reduce and scatter. Finally, allreduce is the addition of the cost of reduce and broadcast.

8 Conclusion

In summary, we discussed latency-optimal solutions for collective communication methods using the Minimum Spanning Tree Algorithm. These methods minimize the number of rounds in which communication happens. However, there is room for improvement as not all nodes have a direct data link between them, and many may remain idle. The next section will cover the Large Message Size Algorithm, which prioritizes bandwidth and utilizes all available data links.