CS633 - PARALLEL COMPUTING

ASSIGNMENT-2

March 30, 2021

The execution command to run the code is python3 run.py

————————————————————————

The Assignment1 directory consists of the following files :

1. Makefile
2. plot.py
3. run.py
4. src.c
5. script.py
6. nodefile.txt
7. plot_BCAST.png
8. plot_GATHER.png
9. plot_REDUCE.png
10. plot_ALLTOALLV.png
11. data_4_2048.txt
12. data_4_32768.txt
13. data_4_262144.txt
14. data_16_2048.txt
15. data_16_32768.txt
16. data_16_262144.txt
17. readme.pdf

The 6 data files correspond to the times taken by the nodes(4 and 16) for the respective doubles(16KB = 2048 , 256KB= 32768 ,2048KB=262144) for ppn=1 and 8 , i.e. data_4_2048.txt consists of data with 4 processes and doubles = 16 KB (16*1024/8 = 2048) , wherein the first 8 entries correspond to ppn=1 , with first 2 entries are for Bcast default and Bcast optimised , next two for Reduce default and Reduce Optimised , next two for Gather default and Gather reduce and next two for AlltoAllv default and AlltoAllv optimised. Then the next 8 entries similarly correspond to the default and optimised versions for ppn=8 . This this then repeated for 10 iterations and hence data file will have 160 entries.

The script.py file generates the hostfile in such a manner that it first checks which hosts are up from each group and then we are choosing maximum number of nodes which are live in all of the groups (i.e. equal no of nodes from each group) which are in power of 2 and placing

them in consecutive manner. This ensures that the each leader process will have same amount of data to communicate to other leaders.

We performed the following optimisations for the various algorithms ::

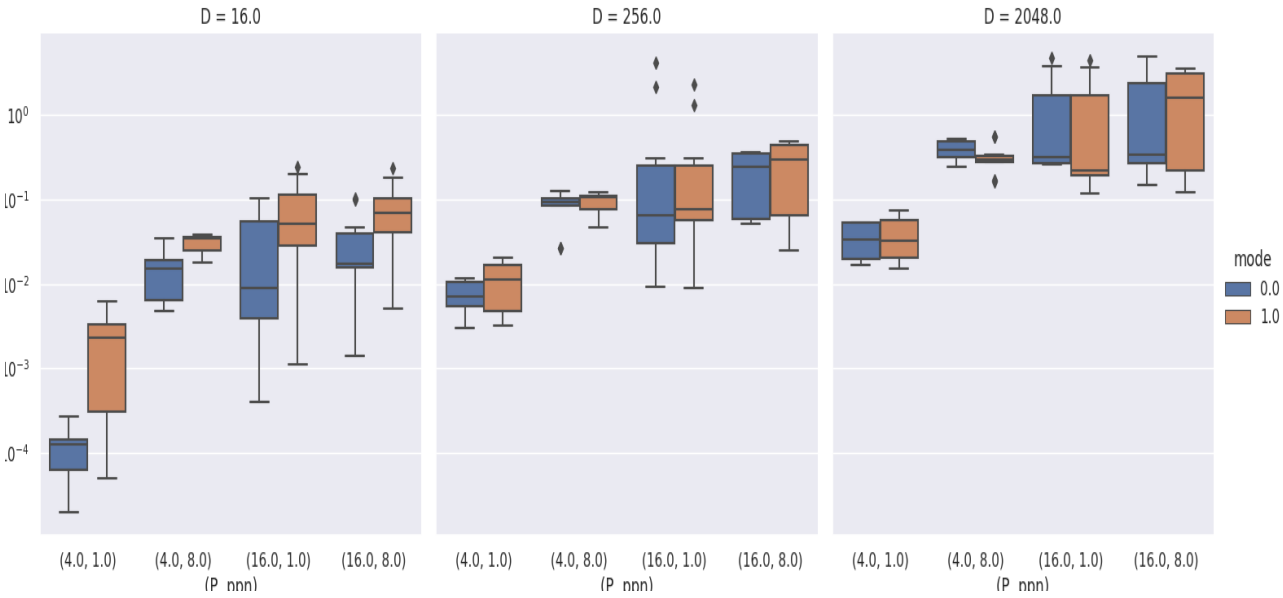
I. Our main Optimisation ::

If ppn=1 and if all the nodes are allocated to only one group , we are not applying any optimisation and calling the default call only as it doesn't make sense to make communicators and leaders among them since they only belong to the same group and this only add overhead due to the creation of communicators. Also , for the case of ppn=8 , if all the processes are on the same node , we are only using the default collective call only due to the same reason as stated above.

1. GATHER : We created an intranode communicator , intragroup communicator and intergroup communicator. The intranode communicator would **GATHER** the data from the processes present on its node . The intragroup communicator will **GATHER** the data onto the root process (Here we have considered 0 to be the root process) , in this case the communication will happen only between the leaders of the respective nodes present in the same group and hence **Only one collective call per node** will take place. Now , after this , the leader process will now be choosen from each of the groupes (leader processes will be those which have the data gathered from the rest of the nodes of the same group) , and then an inter-group communication , will take place between these inter-group leaders , resulting to gathering of all the data to one of the leaders, (which in our case we considered to be 0). Hence , at the end 0 will have all the data on it.

The plots are as shown ::

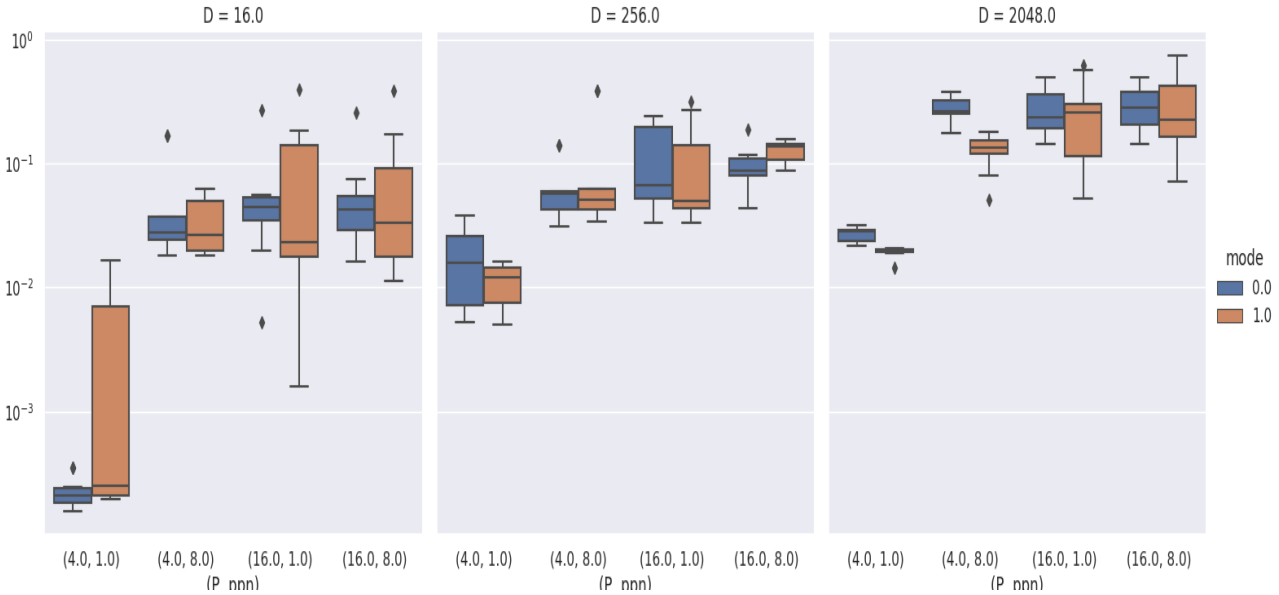
"GATHER"



2. BCAST : Here also similar to the above case, we created intranode communicator , intra-group communicator and intergroup communicator. The intergroup communicator was used to broadcast the value from the root process 0 to all the leaders of each of group . Then the intragroup communicator was used to broadcast the value from the leader processes of the respective groups , to all the nodes present in the group as that of 0. Then each of the leader process will perform intra-node Bcast to provide the processes present in the same node with the required data using intra-node communicator.

The plots are as shown ::

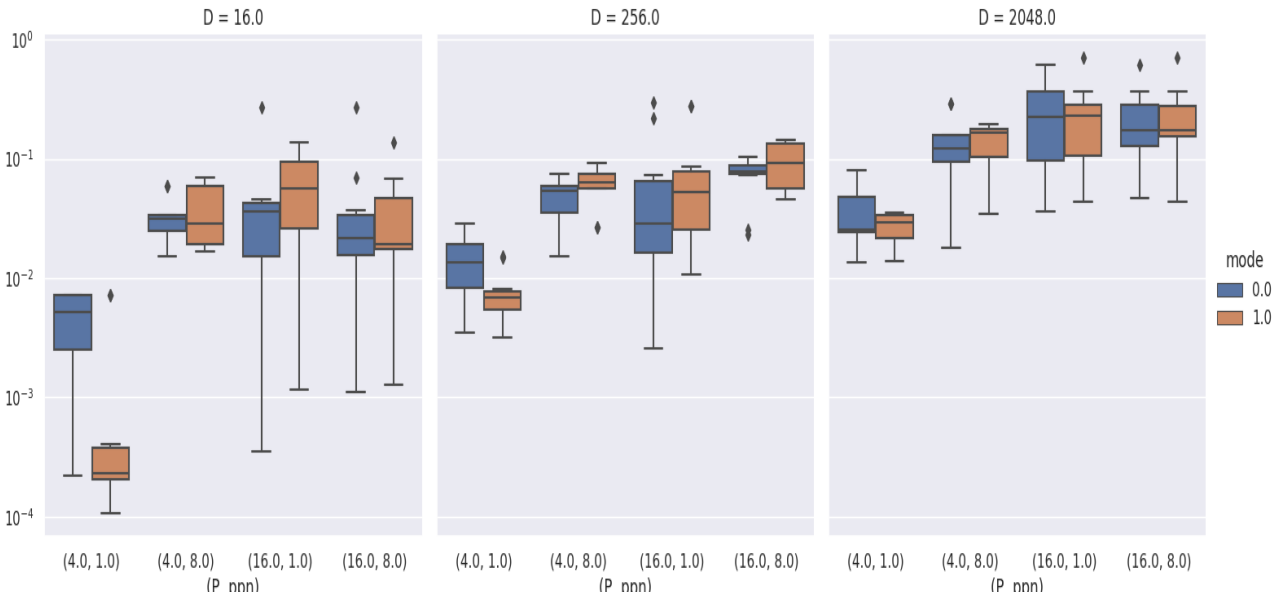
"BCAST"



3. REDUCE : For this case also , we created intranode , intragroup and intergroup communicators. The data from each process in the same node is provided to the leader of that node with **REDUCE** operation . Then the data from each node on the same group is sent to the leader of that group and then with the help of inter-group communication , the data from the leader process of all the groupes it transferred to the root process(0).

The plots are as shown ::

"REDUCE"



4. ALLTOALLV : In case of alltoallv also , just like the above cases, we created intranode communicator , intragroup communicator and intergroup communicators. For the communication part to happen , first we communicated between the processes how much data will they all receive , which was done using a collective call of AlltoAll , which distributed the amount of data to every process , which it is going to receive.

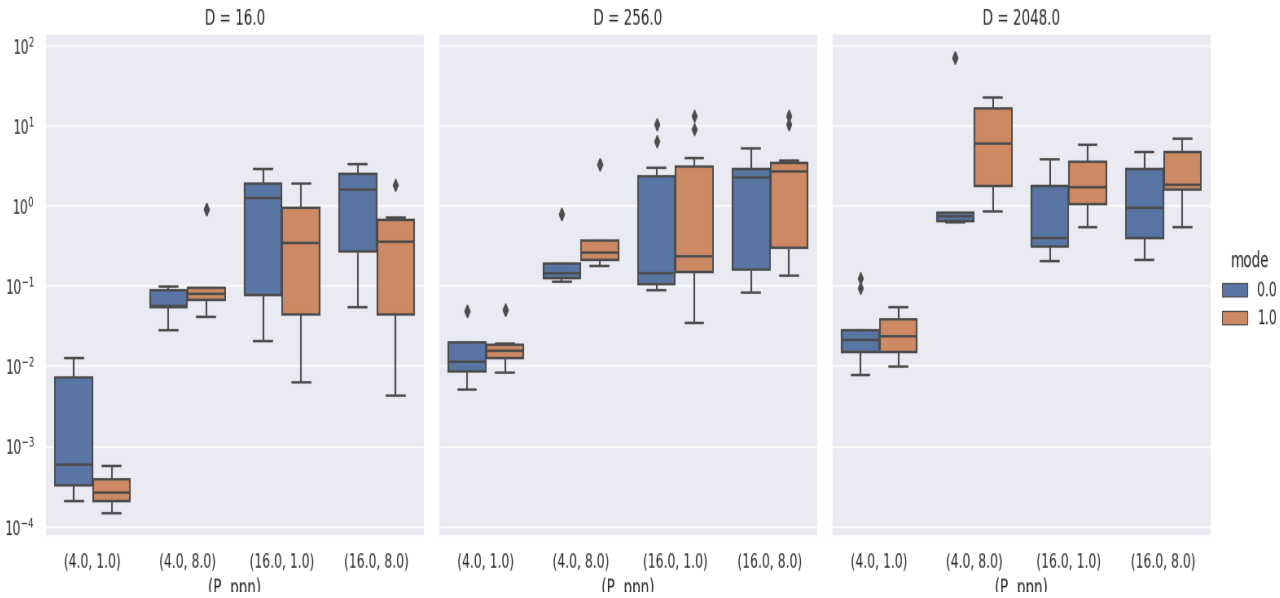
In case of ppn = 1 ,we considered inter-group communication , wherein with the usage of AlltoAll between every process in the node , it was known that what amount of data , every process wants to communicate and accordingly the sendcount and senddisplacement arrays were created and then AlltoAllv was performed for actual message transfer between the processes. Then after this , all the data was gathered from every process present in the same group , using Gatherv , by which now the leader of each group has the required data it wants to send to other processes present in the different group.This data , after getting received by usage of Gatherv call, was re-arranged in order to make it appropriate for sending via AlltoAllv call . Then , AlltoAllv was performed via inter-group communicator and now the leader of each group has the data it wants from every other group leader. Now , this data is rearranged to make it suitable for sending via Scatterv collective call. This data is then transferred to every process via the scatterv collective call so that now every node of the group has the required data it wants.
So , to summarize we can say , that there was firstly intra-group communication to gather the data to leader process and the inter-group communication to distribute among leaders and then again intra-group communication to distribute amongst all the nodes of the same group.

In case of ppn = 8 ,we considered intra-node communication , wherein with the usage of AlltoAllv between every process in the node , every process of each node will have each others required data . Then with the usage of Gatherv , it was known that what amount of data , every process wants to communicate and accordingly the sendcount and senddisplacement arrays were created and then AlltoAllv was performed for actual message transfer between the processes. This data was re-arranged in order to make it appropriate for sending via AlltoAllv call . Then , AlltoAllv was performed via inter-node communicator and now the leader of each node has the data it wants from every other node leader. Now , this data is rearranged to make

it suitable for sending via Scatterv collective call. This data is then transferred to every process via the scatterv collective call so that now every process of each node has the required data it wants. So , to summarize we can say , that there was firstly intra-node communication to collect the data , followed by inter and intra-group communication to distribute among leaders and then again intra-node communication to distribute amongst all the inner processes of one node.

The plots are as shown ::

"ALLTOALLV"



As the hostfile generating the nodes from each group and putting in consecutive manner cost for creating groups and respective communicator for intra-node, intra-switch and inter-switch communication is higher than calling individual collective calls by each process if the data size is low and hence we are not getting much optimisations. In case of large data size communication , if there are single calls from each process , this will increase the congestion in the network as there will be multiple intra and inter communication happening and hence , we decided to create leaders , so that we can reduce multiple single communications to single leader communication and hence , we got some optimisations for some configurations for the optimised version of code, as for some cases , creating sub-communicators will not overshadow the network congestion part, as visible in the case of AlltoAllv , as we need to re-arrange the data in order to call the default collective call.

II. Other Optimisations ::

For the case of gather and reduce , for small data size , we tried doing only intra node and inter-group communication (because we weren't getting very good optimisations , so thought of reducing the cost of creating intra-group communicators) , i.e. making a process leader out of every node present , and then perform communication between them . But this approach also did not provide any optimisation for our case.

After that in order to perform alltoallv , we tried to use scatterv also , wherein each rank will be calling the scatterv collective call as root, to scatter their data among all the other processes but on the basis of the plots generated , we concluded that it did not give enough optimisation and performed worse in comparison to the earlier optimisation that was performed.

Next , we also tried to work upon another optimisation . For this optimisation , we generated the hostfile in such a manner , that we are first finding what live hosts are up from each group and then randomly putting all the live hosts in the hostfile.
In this optimisation , we will be finding the host name for a particular rank via MPI_Get_processor_name and then finding which of them are in the same group with the help of nodefile.txt and then we can share this information among all the processes using AllGather collective call so that now every rank will know which ranks belong to the same group with itself . After that we perform intra-group communication with those that are in the same group and then chose a leader from each intra-communication group and perform inter-group communication with those leaders present in the different groups. The leader processes will use vector collective calls (e.g. gatherv in case of gather) since randomise the placement of processes in the group and hence there are unequal number of processes in each group.We were able to implement this approach for gather, bcast and reduce collective calls but not for alltoallv , since it was quite complicated to deal with unequal number of nodes in each group for that case.

We thought that this approach is utilising the topological information and theoretically should provide the better optimisation than the default one as it would use lesser number of intergroup communication when compared to the default one.

We weren't able to submit this optimisation as the final approach as although the optimised versions of all the three collective calls worked but AlltoAllv gave some complicated errors as it was working for ppn=1 for all the available data sizes but for ppn=8 it gave error for data size 2048KB.

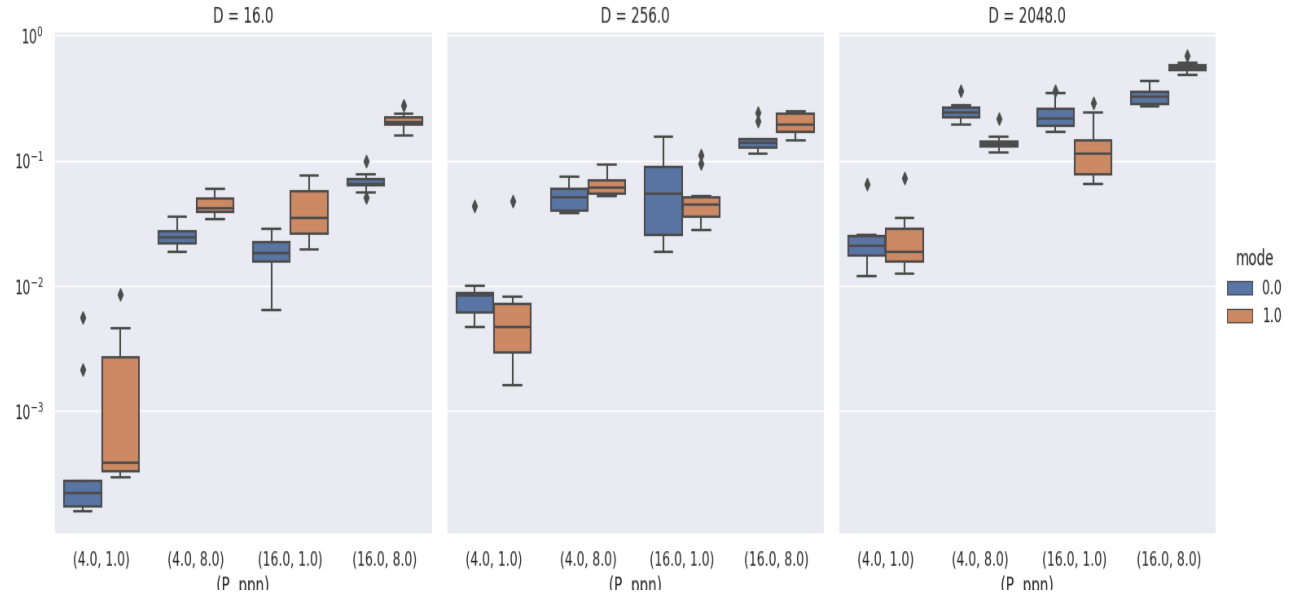
The error was ::
Fatal error in PMPI_Alltoallv: Invalid buffer pointer, error stack:
PMPI_Alltoallv(389):
MPI_Alltoallv(sbuf=0x7f6447a9a010,
scnts=0x7f6505a83f60, sdispls=0x7f6505a83160,
MPI_DOUBLE, rbuf=(nil), rcnts=0x7f6505a83100,
rdispls=0x7f6505a83120, datatype=MPI_DOUBLE,
comm=comm=0xc4000007) failed
PMPI_Alltoallv(360): Null buffer pointer

Even though we were allocating the rbuf array dynamically, still it showing null buffer pointer . But if the same code is run such that only 12 core nodes are selected , then it worked out fine. Hence , we weren't able to debug how this happened.

The following are the plots we got for this particular optimisations we performed :

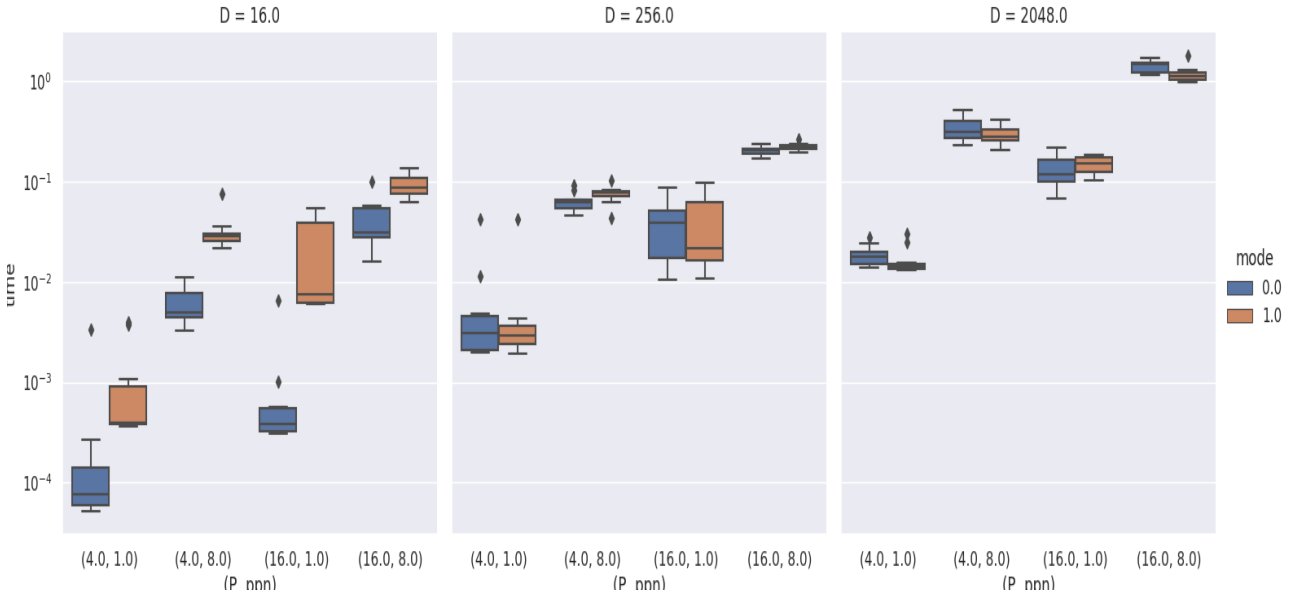
The plots are as shown ::

"Bcast"



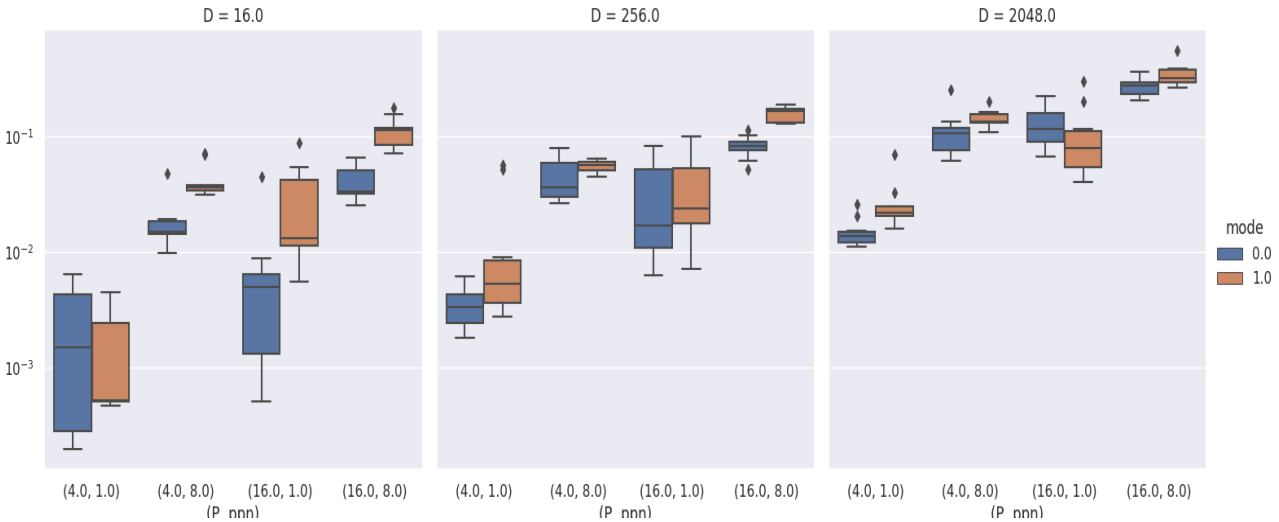
The plots are as shown ::

"Gather"



The plots are as shown ::

"Reduce"


D = 16.0
D = 256.0
D = 2048.0
10^{-1}
10^{-2}
10^{-3}
(4.0, 1.0)
(4.0, 8.0)
(16.0, 1.0)
(16.0, 8.0)
(P, ppp)
mode
0.0
1.0