Если пойдем через отправку сообщений (Bcast), то выглядеть будет так:

0 поток отправляет всем данные, 1 поток отправляет всем данные, 2 поток отправляет всем данные, <...>, n-ный поток отправляет всем данные. И так M раз.

И не забывайте, цикл и в том, и в другом случае должен работать так, чтобы **все** процессы вызывали коллективную операцию, просто от конфигурации зависит, собирает этот процесс данные или отправляет (Bcast), или отправляет данные или суммирует (Reduce).

Давайте еще раз: у Вас вариант с кольцом, то есть 0 процесс передает сообщение 1, 1 передает 2, 2 передает 3 и т.д.

Соответственно тут реализация будет такой:

Из 0 процесса отправляем сообщение всем, потом из 1 отправляем всем, далее из 2 отправляем всем и т.д. Это если делать через Bcast.

Если делать через Reduce, то:

Из 0 процесса собираем ото всех, потом из 1 собираем ото всех, далее из 2 собираем ото всех и т.д.

Вся сложность этой лабы состоит только в том, чтобы собрать сообщения (Bcast) или отправить (Reduce).

Т.е. одна из этих команд в зависимости от варианта должна быть вызвана **всеми** процессами, т.к. управляющий (т.е. рассылающий или собирающий ) процесс определяется при вызове как раз этих команд.

И повтор такой рассылки должен быть M раз. То есть M раз мы пробегам рассылками или сборами по процессам от 0 до N-ного процесса.

С рассылкой уже хорошо, поняли, как работает.

Теперь давайте поправим структуру вокруг Bcast.  
  
Сейчас Вы храните дополнительные, в общем-то, не нужные данные - информацию о прошлом процессе и следующем. При этом цикл у Вас один и по M, то есть по числу раз, которое мы должны пробежаться по кольцу.  
  
Давайте попробуем упростить.

Есть формула с взятием по mod, то есть берем остаток от деления. В данном случае он поможет Вам сделать все в одном цикле, если вложенные Вас не устраивают. Тогда цикл будет по M\*ProcNum, а считать номер процесса, откуда вещаем будем по формуле: proc = (i)%ProcNum; А вещание делаем как раз по этой величине proc.

|  |  |
| --- | --- |
| |  | | --- | |  | |

Нет, снова не совсем попали...

У Вас сам цикл такой должен быть: for (int i = 0; i < M\*ProcNum; i++)  
a = ((a + 1) % ProcNum);

И вывод от printf же делается не абсолютно на каждой итерации? У Вас в нынешней реализации существуют лишние срабатывания. Если сделаете не char рассылаемое сообщение, а int и будете в рассылающем(только в рассылающем процессе) делать +1, то увидите это. Поправляйте.

У Вас же 2 повтора и 4 процесса. Почему повторов 4 тогда? Должно же быть так: 0, 1, 2, 3, 0, 1, 2, 3. И все. Только 2 раза

#include <stdio.h>

#include <mpi.h>

#include <iostream>

void main(int argc, char\* argv[])

{

int M = 5;

int ProcRank, ProcNum, prev, next;

int a = 0;

int start = 0;

int rank;

int message = 0;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

for (int i = 0; i < M \* ProcNum; i++)

{

if (ProcRank == 0 && start == 0)

{

message ++;

//MPI\_Send(message, strlen(message) + 1, MPI\_CHAR, 1, 0, MPI\_COMM\_WORLD);

//printf("%s process rank %d send message to 1 \n", message, ProcRank);

}

if (start == 0)

{

start = 1;

MPI\_Bcast(&message, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

//if (a == ProcNum) a = 1;

a = (a + 1) % ProcNum;

prev = ProcRank - 1;

next = ProcRank + 1;

if (ProcRank == 0) prev = ProcNum - 1;

if (ProcRank == ProcNum - 1) next = 0;

if (prev == a)

printf("Proc mumber: %d received message: %d from proc number %d for the %d time\n", ProcRank, &message, a, i);

MPI\_Bcast(&message, 1, MPI\_INT, (a + 1) % ProcNum, MPI\_COMM\_WORLD);

a++;

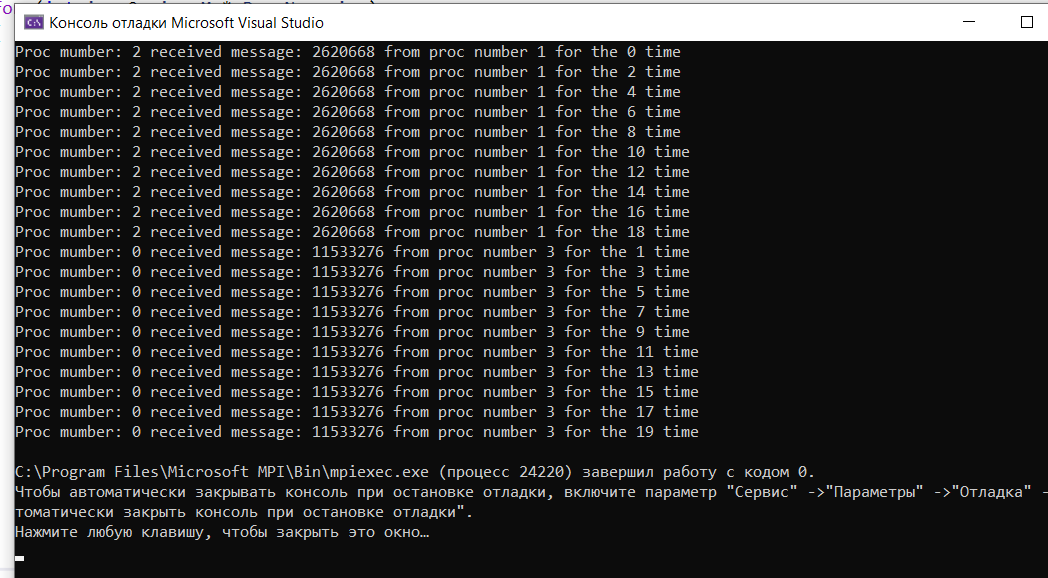
}

//MPI\_Bcast(&message, 1, MPI\_CHAR, 2, MPI\_COMM\_WORLD);

//printf("All processes received message : \"%s\" from processor number %d for the %d time \n\n", /\*ProcRank,\*/ //message, recvFrom, i);

MPI\_Finalize();

}



#include <stdio.h>

#include <mpi.h>

#include <iostream>

int main(int argc, char\* argv[])

{

int M = 2;

int ProcRank, ProcNum;

int a = 0;

int mes = 0;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

for (int i = 0; i < M \* ProcNum; i++)

{

//if (a == ProcNum) a = 1;

a = (a + 1) % ProcNum;

if (ProcRank == a)

mes += ProcRank;

MPI\_Bcast(&mes, 1, MPI\_INT, a, MPI\_COMM\_WORLD);

if (ProcRank == a)

printf("Proc mumber: %d send message: %d from proc number %d for the %d time\n", ProcRank, mes, (a) % ProcNum, i);

if (ProcRank == (a + 1) % ProcNum)

printf("Proc mumber: %d recieve message: %d for the %d time\n", ProcRank, mes, i);

}

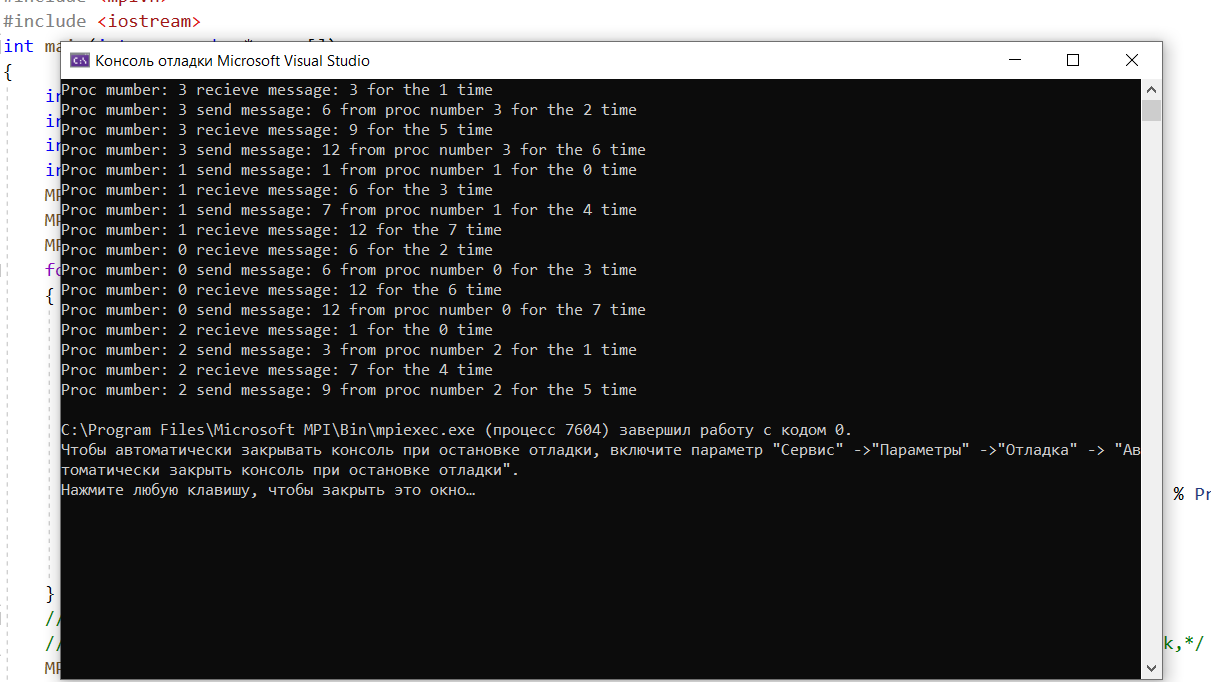
//MPI\_Bcast(&message, 1, MPI\_CHAR, 2, MPI\_COMM\_WORLD);

//printf("All processes received message : \"%s\" from processor number %d for the %d time \n\n", /\*ProcRank,\*/ //message, recvFrom, i);

MPI\_Finalize();

return 0;

}



#include <stdio.h>

#include <mpi.h>

#include <iostream>

void main(int argc, char\* argv[])

{

int M = 5;

int ProcRank, ProcNum, prev, next;

int a = 0;

int start = 0;

int rank;

char message[] = "message";

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

for (int i = 0; i < M \* ProcNum; i++)

{

if (ProcRank == 0 && start == 0)

{

char message[] = "message";

//MPI\_Send(message, strlen(message) + 1, MPI\_CHAR, 1, 0, MPI\_COMM\_WORLD);

//printf("%s process rank %d send message to 1 \n", message, ProcRank);

}

if (start == 0)

{

start = 1;

MPI\_Bcast(&message, strlen(message) + 1, MPI\_CHAR, 0, MPI\_COMM\_WORLD);

}

//if (a == ProcNum) a = 1;

a = (a + 1) % ProcNum;

prev = ProcRank - 1;

next = ProcRank + 1;

if (ProcRank == 0) prev = ProcNum - 1;

if (ProcRank == ProcNum - 1) next = 0;

if (prev == a)

printf("Proc mumber: %d received message: %s from proc number %d for the %d time\n", ProcRank, &message, a, i);

MPI\_Bcast(&message, strlen(message) + 1, MPI\_CHAR, (a + 1) % ProcNum, MPI\_COMM\_WORLD);

a++;

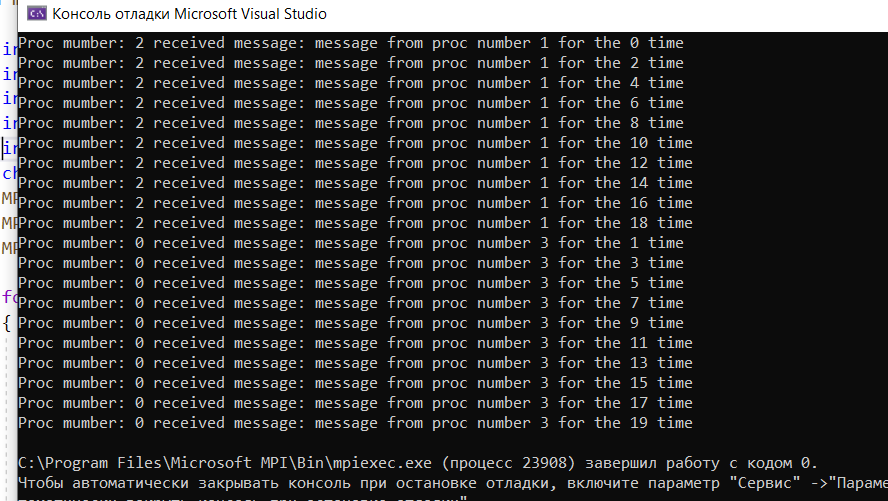
}

//MPI\_Bcast(&message, 1, MPI\_CHAR, 2, MPI\_COMM\_WORLD);

//printf("All processes received message : \"%s\" from processor number %d for the %d time \n\n", /\*ProcRank,\*/ //message, recvFrom, i);

MPI\_Finalize();

}



#include <stdio.h>

#include <mpi.h>

#include <iostream>

int main(int argc, char\* argv[])

{

int M = 2;

int ProcRank, ProcNum;

int a = 0;

char mes[] = "privet";

//int mes = 0;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

for (int i = 0; i < M \* ProcNum; i++)

{

//if (a == ProcNum) a = 1;

a = (a + 1) % ProcNum;

//if (ProcRank == a)

//mes += ProcRank;

MPI\_Bcast(&mes, strlen(mes)+1, MPI\_CHAR, a, MPI\_COMM\_WORLD);

if (ProcRank == a)

printf("Proc mumber: %d send message: %s from proc number %d for the %d time\n", ProcRank, mes, (a) % ProcNum, i);

if (ProcRank == (a + 1) % ProcNum)

printf("Proc mumber: %d recieve message: %s for the %d time\n", ProcRank, mes, i);

}

//MPI\_Bcast(&message, 1, MPI\_CHAR, 2, MPI\_COMM\_WORLD);

//printf("All processes received message : \"%s\" from processor number %d for the %d time \n\n", /\*ProcRank,\*/ //message, recvFrom, i);

MPI\_Finalize();

return 0;

}

