1. **Базові поняття MPI: процес, ранг, комунікатор, особливості написання MPI програми.**

**Процесс** - совокупность взаимосвязанных и взаимодействующих действий, преобразующих входящие данные в исходящие.

**Коммуникатор** представляет собой структуру, содержащую либо все процессы,

исполняющиеся в рамках данного приложения, либо их подмножество.

Процессы, принадлежащие одному и тому же коммуникатору, наделяются

общим контекстом обмена. Операции обмена возможны только между

процессами, связанными с общим контекстом, то есть, принадлежащие одному

и тому же коммуникатору (рис. 13). Каждому коммуникатору присваивается

идентификатор. В MPI есть несколько стандартных коммуникаторов:

* MPI\_COMM\_WORLD–включаетвсепроцессыпараллельнойпрограммы;
* MPI\_COMM\_SELF–включаеттолькоданныйпроцесс;
* MPI\_COMM\_NULL– пустой коммуникатор, не содержит ни одного

процесса.

В MPI имеются процедуры, позволяющие создавать новые коммуникаторы,

содержащие подмножества процессов.

**Ранг** процесса представляет собой уникальный числовой идентификатор, назначаемый процессу в том или ином коммуникаторе. Ранги в разных коммуникаторах назначаются независимо и имеют целое значение от 0 до число процессов – 1.

1. **Базові поняття MPI: блокуючий та неблокуючий обмін повідомленнями.**

Сообщение содержит пересылаемые данные и служебную информацию. Для того, чтобы передать сообщение, необходимо указать:

* ранг процесса-отправителя сообщения;
* адрес, по которому размещаются пересылаемые данные процесса-отправителя;
* тип пересылаемых данных;
* количество данных;
* ранг процесса, который должен получить сообщение;
* адрес, по которому должны быть размещены данные процессом-получателем.
* тег сообщения;
* идентификатор коммуникатора, описывающего область взаимодействия, внутри которой происходит обмен.

Тег это задаваемое пользователем целое число от 0 до 32767, которое играет роль идентификатора сообщения и позволяет различать сообщения, приходящие от одного процесса. Теги могут использоваться и для соблюдения определенного порядка приема

сообщений.

Прием сообщения начинается с подготовки буфера достаточного размера. В этот буфер записываются принимаемые данные. Операция отправки или приема сообщения считается завершенной, если программа может вновь использовать буферы сообщений.

Двухточечные обмены используются для организации локальных и неструктурированных коммуникаций.

При выполнении глобальных операций используются коллективные обмены. Асинхронные коммуникации реализуются с помощью запросов о получении сообщений. Имеется несколько разновидностей двухточечного обмена.

* Блокирующие прием*/*передача приостанавливают выполнение процесса на время приема сообщения.
* Неблокирующие прием*/*передача выполнение процесса продолжается в фоновом режиме, а программа в нужный момент может запросить подтверждение завершения приема сообщения.

1. **Базові поняття MPI: редукція, групові операції.**

В рассматриваемой программе после входной стадии каждый процессор выполняет по существу те же самые команды до заключительной стадии суммирования. Поэтому, если функция f(x) дополнительно не усложнена (т. е. не требует значительной работы для оценки интеграла по некоторым частям отрезка [a;b], то эта часть программы распределяет среди процессоров одинаковую нагрузку. В заключительной стадии суммирования процесс 0 еще раз получает непропорциональное количество работы.

Возможны также варианты организации передач, как и в случае рассылки входных данных. Поэтому следует использовать другие механизмы, более оптимизированные для этой цели.

"Общая сумма", которую нужно вычислить представляет собой пример общего класса коллективных операций коммуникации, называемых операциями редукции. В глобальной операции редукции, все процессы (в коммуникаторе) передают данные, которые объединяются с использованием бинарных операций. Типичные бинарные операции - суммирование, максимум и минимум, логические и т.д. MPI содержит специальную функцию для выполнения операции редукции:

Операция приведения, результат которой передается одному процессу:

int MPI\_Reduce(void \*buf, void \*result, int count, MPI\_Datatype datatype, MPI\_Op op, int root, MPI\_Comm comm)

Входные параметры:

* buf - адрес буфера передачи;

• count - количество элементов в буфере передачи;

• datatype - тип данных в буфере передачи;

• op - операция приведения;

• root - ранг главного процесса;

• comm - коммуникатор.

MPI\_Reduce применяет операцию приведения к операндам из buf, а результат каждой операции помещается в буфер результата result. MPI\_Reduce должна вызываться всеми процессами в коммуникаторе comm, а аргументы count, datatype и op в этих вызовах должны совпадать.

|  |  |
| --- | --- |
| Название операции | Смысл |
| MPI\_MAX | Максимум |
| MPI\_MIN | Минимум |
| MPI\_SUM | Сумма |
| MPI\_PROD | Произведение |
| MPI\_LAND | Логическое И |
| MPI\_BAND | Битовое И |
| MPI\_LOR | Логическое ИЛИ |
| MPI\_BOR | Битовое ИЛИ |
| MPI\_LXOR | Логическое исключающее ИЛИ |
| MPI\_BXOR | Битовое исключающее ИЛИ |
| MPI\_MAXLOC | Максимум и его расположение |
| MPI\_MINLOC | Минимум и его расположение |

Существует также возможность определения собственных операций.

Таким образом, завершение программы вычисления интеграла будет следующим:

/\* Суммирование результатов от каждого процесса \*/

MPI\_Reduce(&integral, &total, 1, MPI\_FLOAT,

    MPI\_SUM, 0, MPI\_COMM\_WORLD);

/\* Вывод результата \*/

Следует отметить, что каждый процессор вызывает MPI\_Reduce() с одинаковыми аргументами. Например, если total имеет значение только для процесса 0, каждый процесс тем не менее использует этот аргумент.

Глобальные операции редукции

* Сумма
* Произведение
* Минимум
* Максимум
* Битовые операции

«Собираемая» переменная может быть простой или массивом.