СЛОЖНЫЙ ПРИМЕР ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ

ТИПЫ ВЗАИМОДЕЙСТВИЯ

- **Взаимодействия типа точка-точка (POINT-TO-POINT)**
  - **Синхронные взаимодействия (блокирующие)**
  - **Асинхронные (неблокирующие)**
  - Комбинированный send/receive
  - Отправка по готовности (“Ready”-send)
  - Буферизованная отправка (Buffered send)

Взаимодействие ***ТОЛЬКО*** между двумя процессами. Один процесс выполняет процедуру отправки сообщения, другой выполняет соответствующую процедуру получения

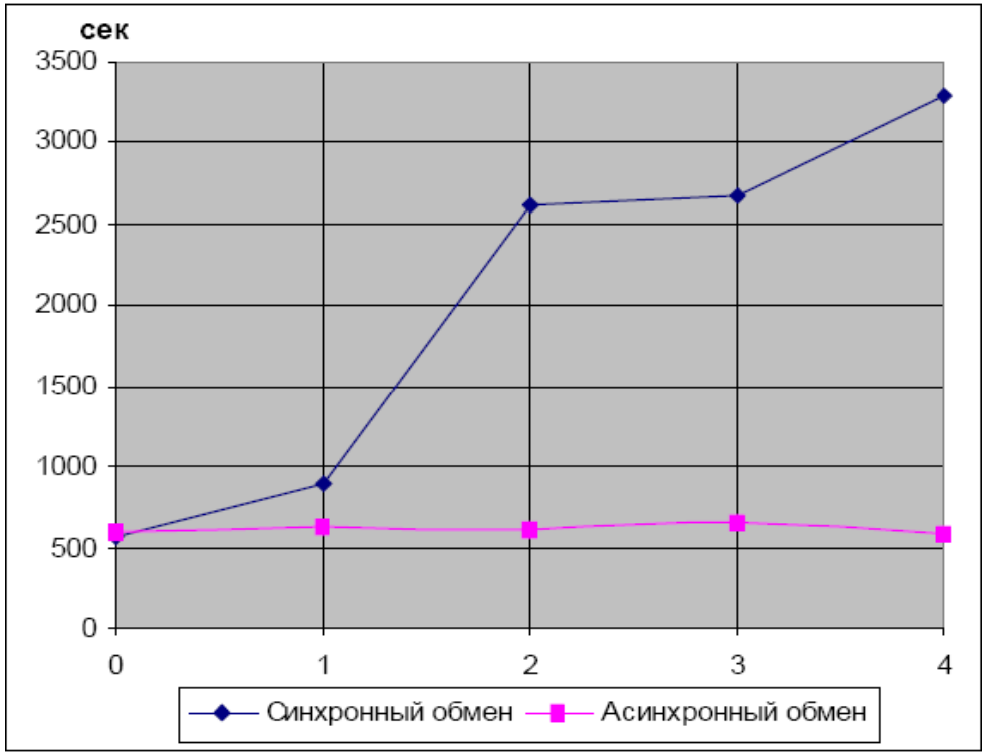
- **Групповые взаимодействия**
  - Групповые операции являются блокирующими
  - Функции не содержат параметра tag
  - Могут быть использованы только с примитивные типы данных

Во взаимодействии принимают участие ***ВСЕ*** процессы, принадлежащие одному и тому же коммуникатору

СИНХРОННЫЕ VS АСИНХРОННЫЕ

Синхронные: Выход из функции синхронной отправки осуществляется только после того, как принимающий процесс подтвердил факт доставки сообщения
Выход из функции синхронного получения производится только после того, как данные получены и готовы для дальнейшего использования

Асинхронные: Выход из функций асинхронной отправки и получения осуществляется «практически» мгновенно.
Пользователь не может быть на 100% уверен в достоверности отправки/получении данных



Зависимость времени синхронных и асинхронных типов обменов от количества процессоров без вычислений (логарифмическая шкала).

СИНХРОННЫЕ ВЗАИМОДЕЙСВИЯ

int
MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

[IN buf]	– адрес передаваемых данных
[IN count]	– количество передаваемых данных
[IN datatype]	– тип передаваемых данных
[IN dest]	– ранг процесса-получателя
[IN tag]	– тег передаваемых данных
[IN comm]	– коммуникатор

int
MPI_Recv(void* buf, int count, MPI_Datatype datatype, int src, int tag, MPI_Comm comm, MPI_Status* status)

[OUT buf]	– адрес буфера принимаемых данных
[IN count]	– количество передаваемых данных
[IN datatype]	– тип принимаемых данных
[IN src]	– ранг процесса-отправителя
[IN tag]	– тег принимаемых данных
[IN comm]	– коммуникатор
[OUT status]	– описание принятого сообщения (содержит ранг отправителя, идентификационный номер tag, количество отправленных элементов)

АСИНХРОННЫЕ ВЗАИМОДЕЙСВИЯ

int
MPI_Isend(void* buf, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm, MPI_Request *request)

[IN buf] – адрес передаваемых данных
[IN count] – количество передаваемых данных
[IN datatype] – тип передаваемых данных
[IN dest] – ранг процесса-получателя
[IN tag] – тег передаваемых данных
[INcomm] – коммуникатор
[OUT request] – идентифицирует коммуникационное событие

int
MPI_Irecv(void* buf, int count, MPI_Datatype datatype, int src, int tag, MPI_Comm comm,
MPI_Request *request)

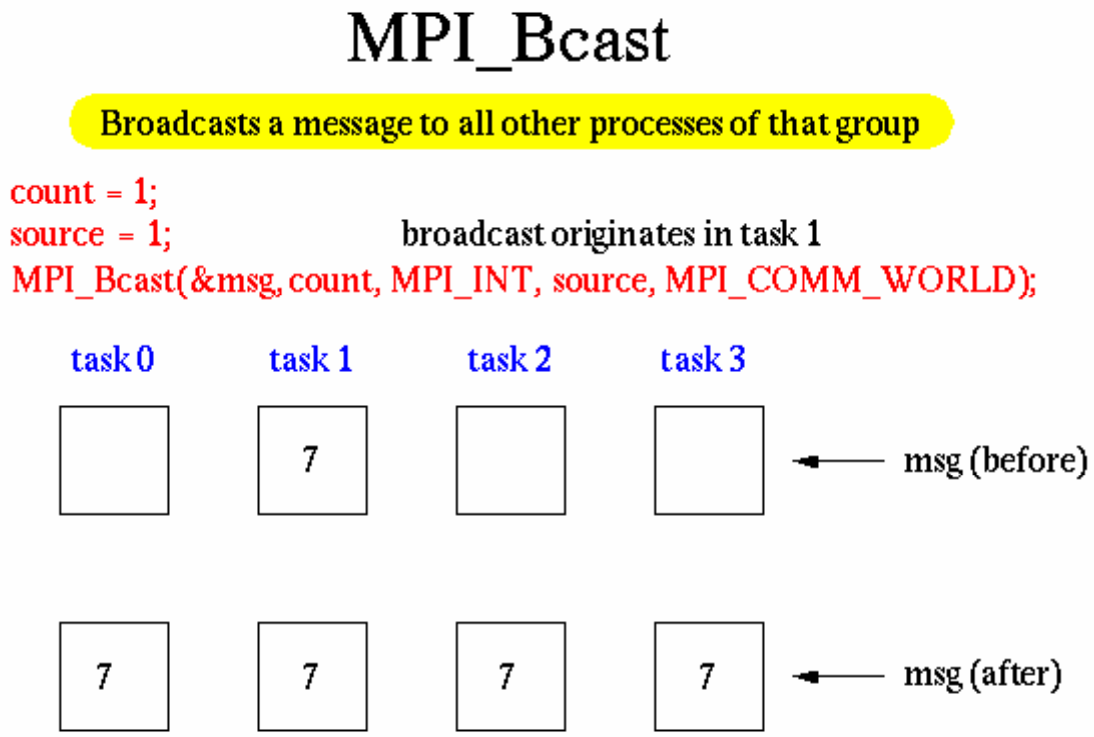
[OUT buf] – адрес буфера принимаемых данных
[IN count] – количество принимаемых данных
[IN datatype] – тип принимаемых данных
[IN src] – ранг процесса-отправителя
[IN tag] – тег принимаемых данных
[INcomm] – коммуникатор
[OUT request] – идентифицирует коммуникационное событие

ПРИМЕР СИНХРОННЫХ/АСИНХРОННЫХ ВЗАИМОДЕЙСТВИЙ

ГРУППОВЫЕ ВЗАИМОДЕЙСТВИЯ - 1

int
MPI_Bcast(void* buf, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

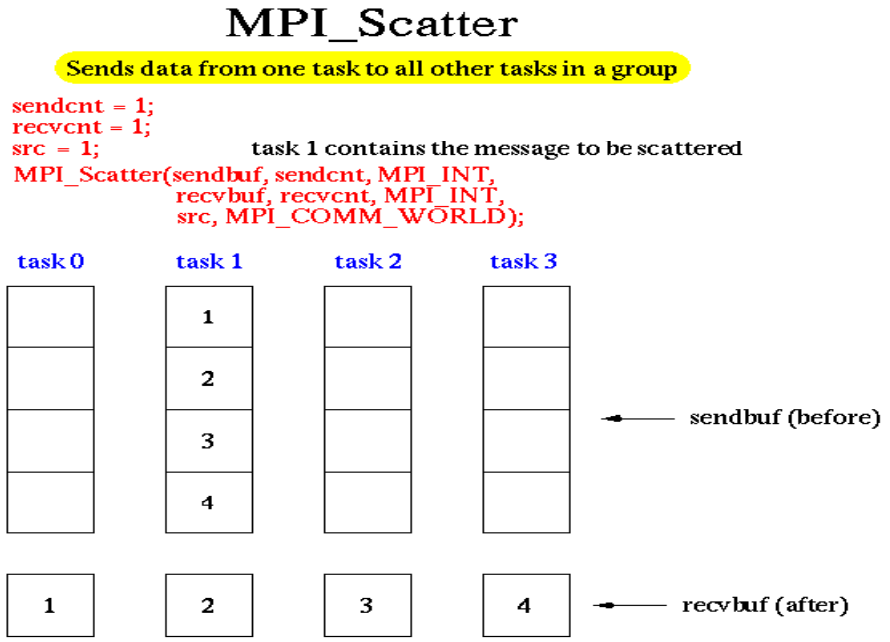
[IN buf]	– адрес передаваемых данных
[IN count]	– количество передаваемых данных
[IN datatype]	– тип передаваемых данных
[IN root]	– ранг процесса-отправителя
[INcomm]	– коммуникатор



ГРУППОВЫЕ ВЗАИМОДЕЙСТВИЯ - 2

int
MPI_Scatter(void* sbuf, int scount, MPI_Datatype sdatatype, void* rbuf, int rcount, MPI_Datatype rdatatype, int root, MPI_Comm comm)

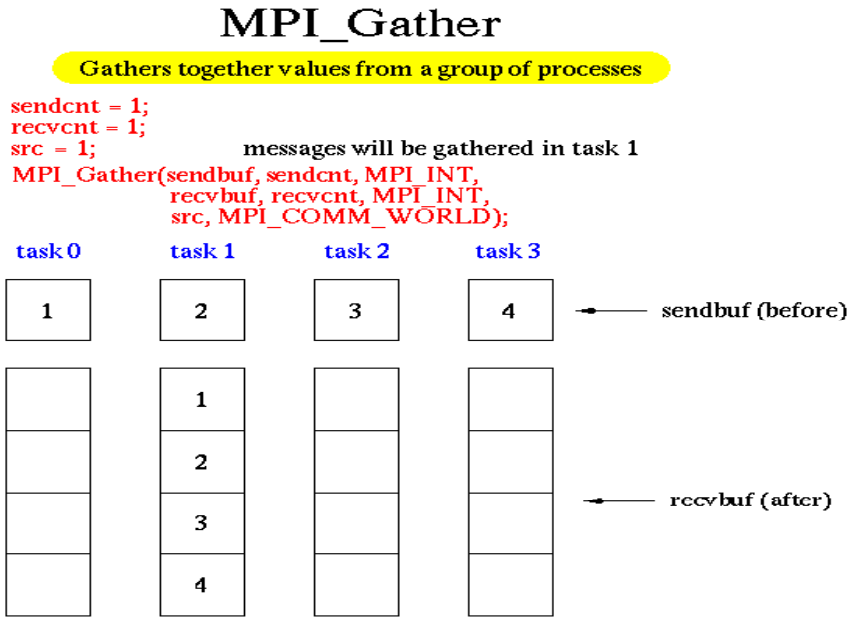
[IN sbuf] – адрес передаваемых данных
[IN scount] – количество передаваемых данных
[IN sdatatype] – тип передаваемых данных
[IN rbuf] – адрес буфера получателя
[IN rcount] – число элементов в буфере получателя
[IN rdatatype] – тип данных получателя
[IN root] – ранг процесса-отправителя
[INcomm] – коммуникатор



ГРУППОВЫЕ ВЗАИМОДЕЙСТВИЯ - 3

int
MPI_Gather(void* sbuf, int scount, MPI_Datatype sdatatype, void* rbuf, int rcount, MPI_Datatype rdatatype, int root, MPI_Comm comm)

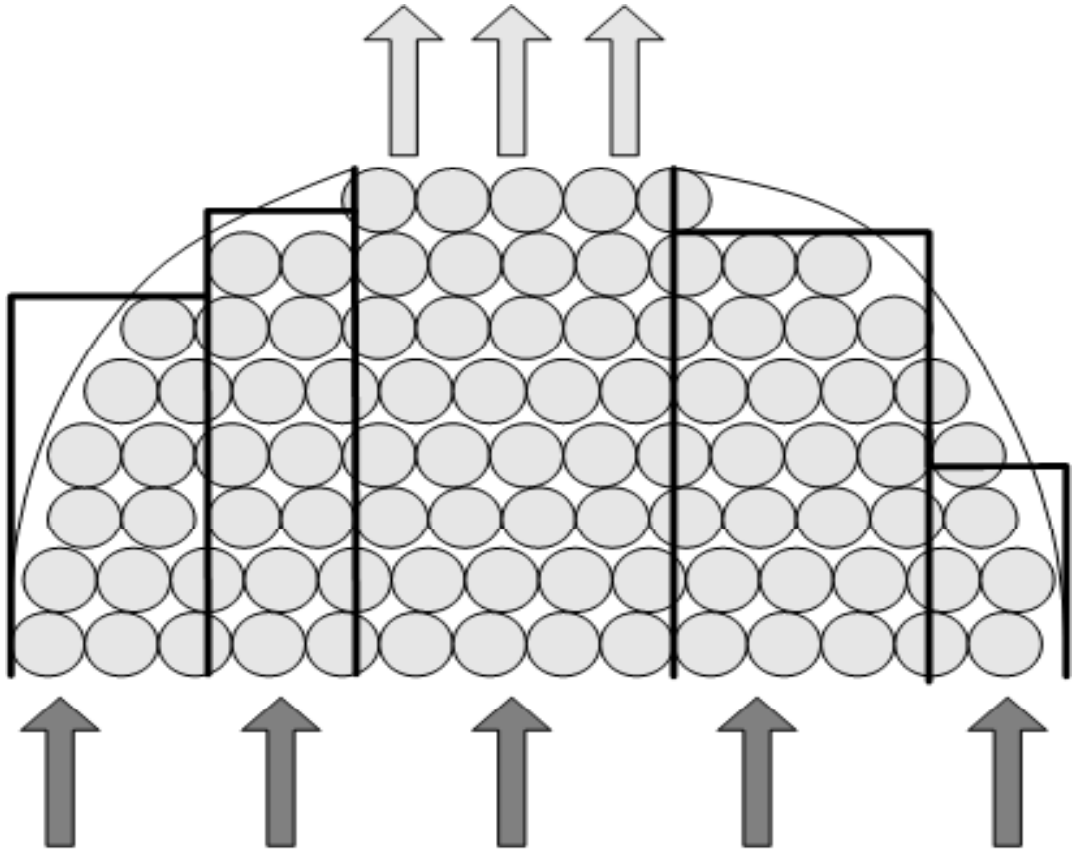
[IN sbuf] – адрес передаваемых данных
[IN scount] – количество передаваемых данных
[IN sdatatype] – тип передаваемых данных
[IN rbuf] – адрес буфера получателя
[IN rcount] – число элементов в буфере получателя
[IN rdatatype] – тип данных получателя
[IN root] – ранг процесса-отправителя
[INcomm] – коммуникатор



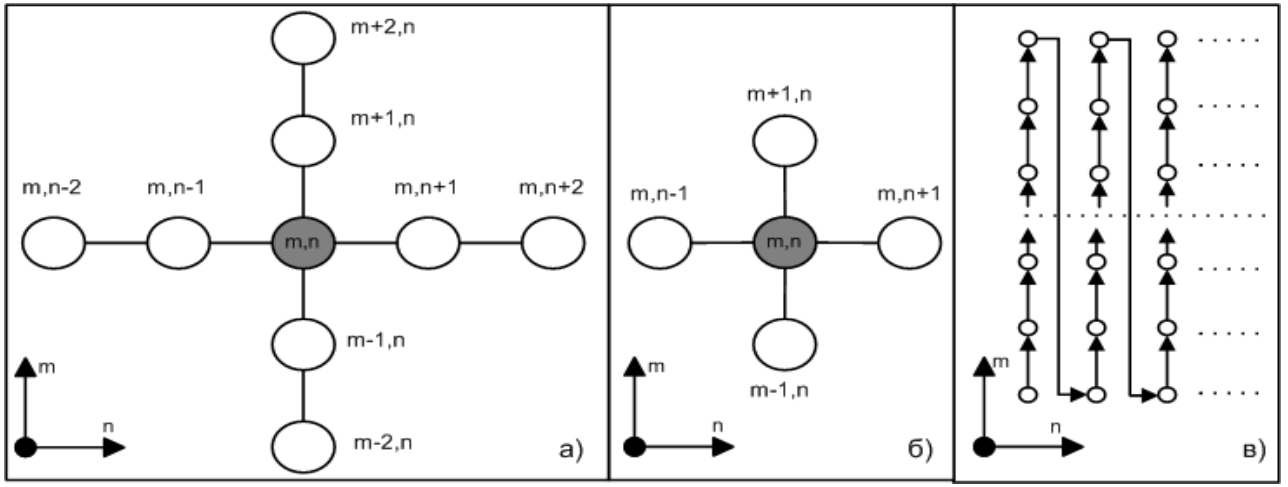
ПРИМЕР ГРУППОВЫХ ВЗАИМОДЕЙСТВИЙ

ПРИМЕР ПРОГРАММЫ «ИЗ ЖИЗНИ»

Программа моделирует двумерное нестационарное движение газа через пористый тепловыделяющий элемент



Схематическое представление основных вычислений



1. Вычисления значений во внутренних узлах сетки происходит послойно, т.е. в вычислениях используются значения предыдущего слоя;
2. Схема вычислений значений температуры газа, горизонтальной и вертикальной скоростей фильтрации представлена шаблоном на рисунке а);
3. Схема вычислений значений температуры твёрдой фазы – шаблон на рисунке б);
4. На рисунке в) показано направление обхода вычислений для зон;
5. Для вычисления значений давления по координате m применяется прямая или обратная прогонки в зависимости от типа зоны – открытой или закрытой соответственно;
6. Вычисление значений в граничных узлах сетки осуществляется с использованием соседних значений текущего слоя.

Технология распараллеливания

Классический сценарий параллельной программы:

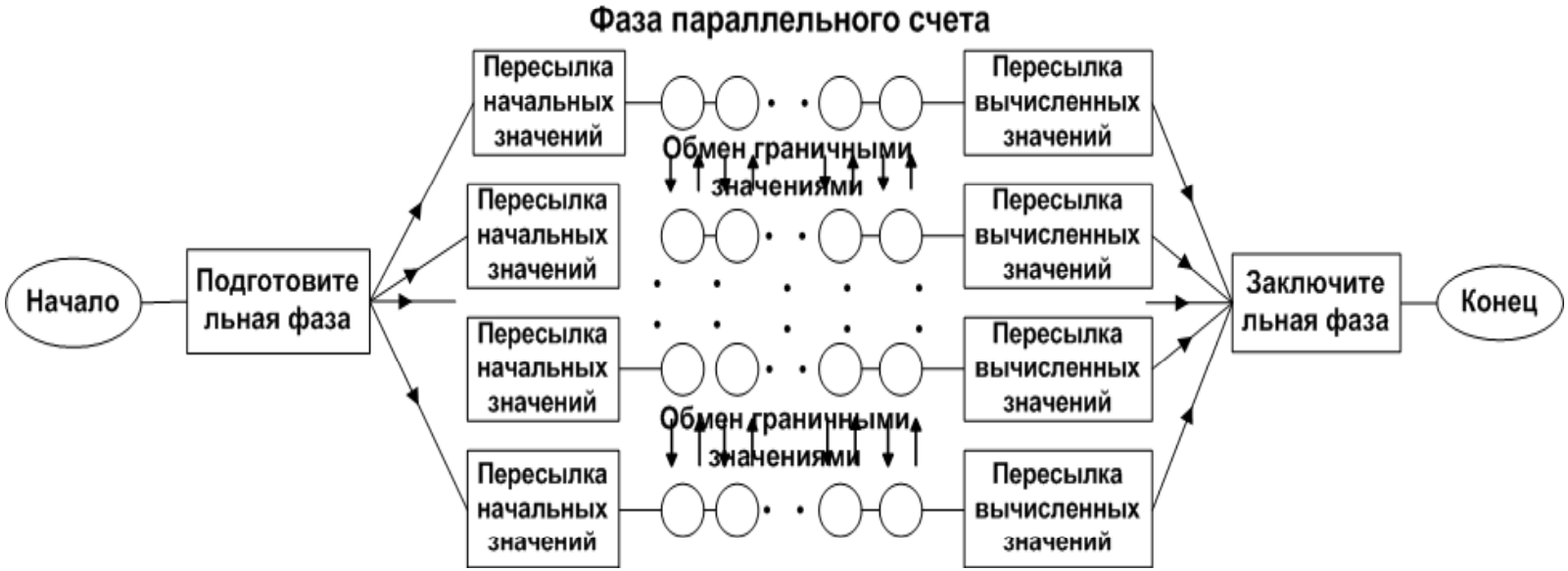
Необходимо ответить на вопросы:

1. Выделить подготовительную, заключительную и фазу параллельного счёта;
2. Выделить минимальную гранулу параллелизма;
3. Определить порядок взаимодействия вычислительных узлов;
4. Произвести балансировку вычислительной нагрузки узлов;
5. Определить примерную оценку оптимального количества процессоров.

Время работы программы в зависимости от количества процессоров описывается формулой:

$$T_{all}(N) = T_{conn}(N) + T_{par}(N) + T_{com}(N)$$

Схема распараллеливания программы

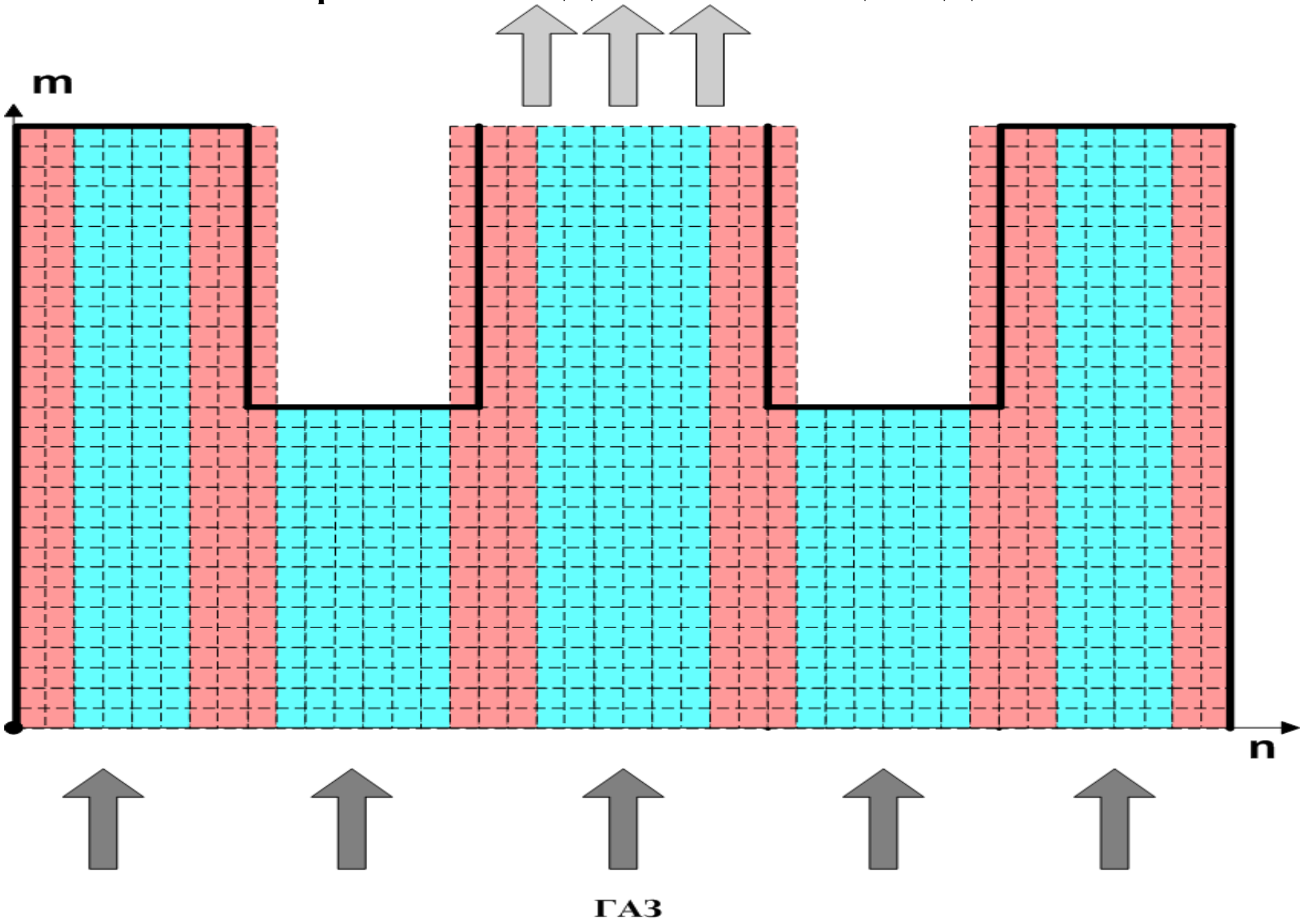


Подготовительная фаза: считывание начальных данных, расчёт используемых в вычислениях переменных, выполнение декомпозиции и рассылки соответствующих данных.

Фаза параллельного счёта: расчёт значений соответствующей каждому узлу области.

Заключительная фаза: Композиция вычисленных значений и запись в соответствующие файлы.

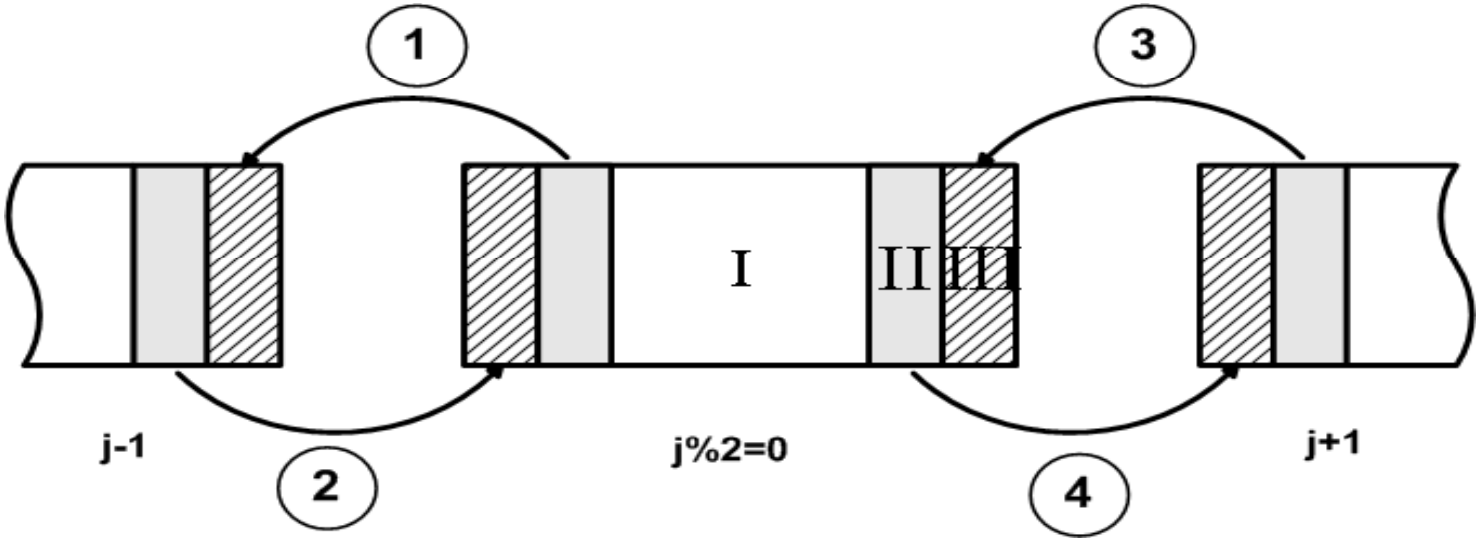
Геометрическая декомпозиция данных



Организация обмена граничными значениями для каждой расчётной области

Необходимо ответить на вопросы:

1. Определить размер граничной области;
2. Выбрать порядок обмена с учётом направления обхода вычислений;
3. Исключить возможность появления ситуации deadlock для вычислительных узлов.



I – значения внутренней области;
II – значения граничной области;
III – значения теневой области

Факторы, влияющие на эффективность программ

Физические

- Тип и характеристики процессора;
- Объём памяти - кэш и оперативной;
- Для параллельной программы:
  - быстродействие коммуникационной среды;
  - количество вычислительных узлов

Структурные

- Структуры данных
- Порядок вычислений на используемых структурах данных

Представление данных в памяти и организация вычислений на них