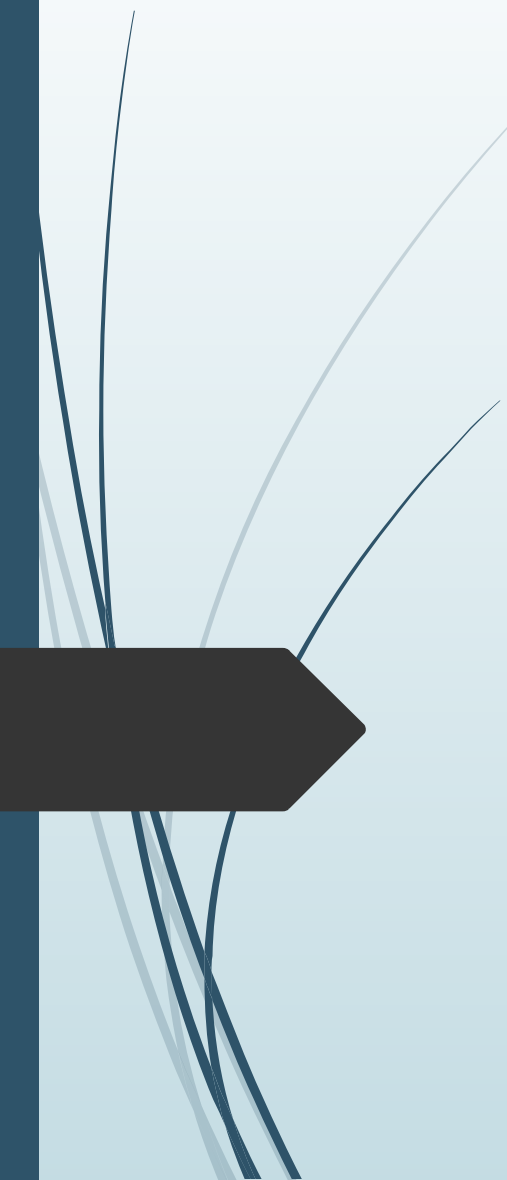
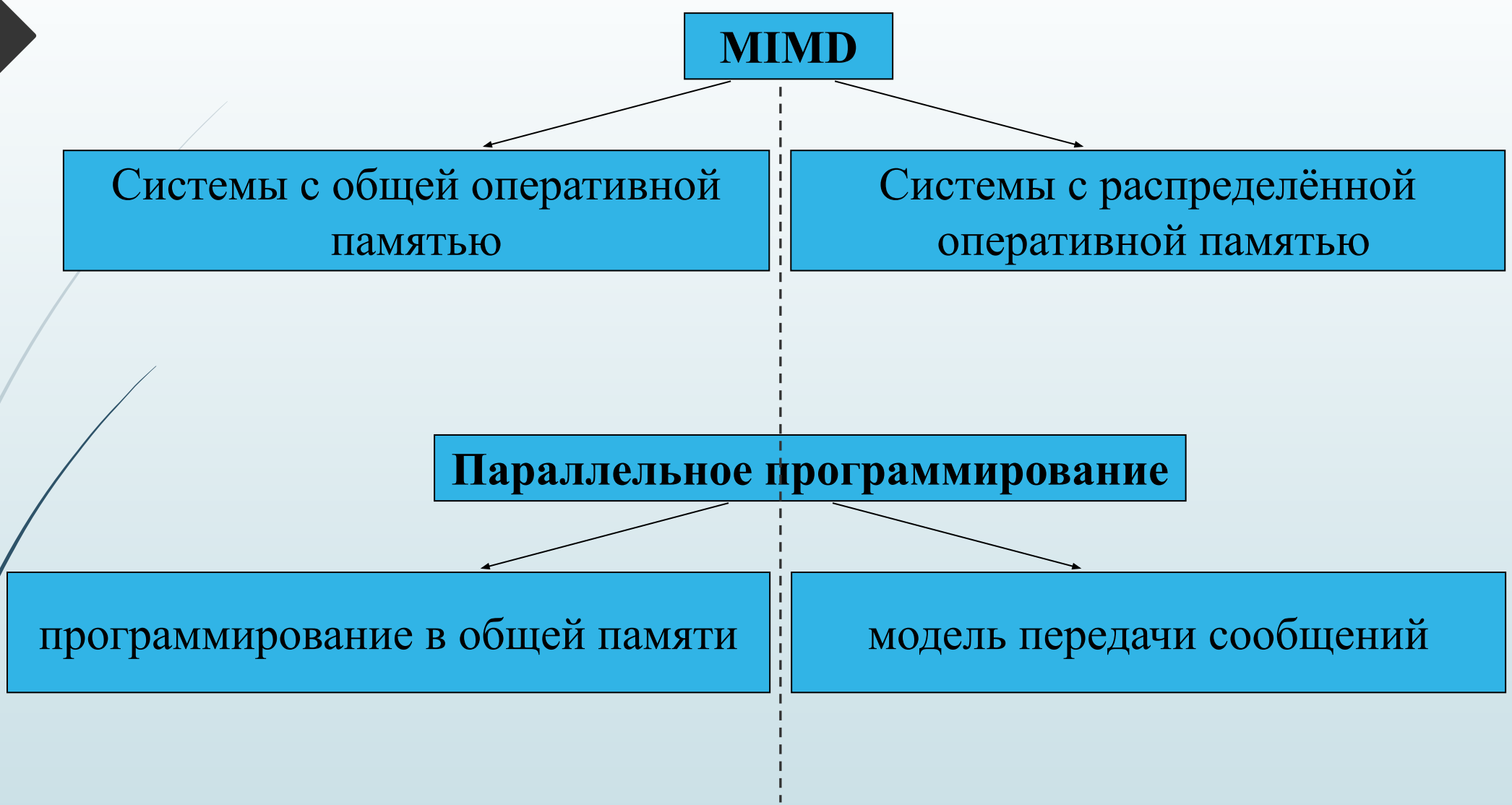
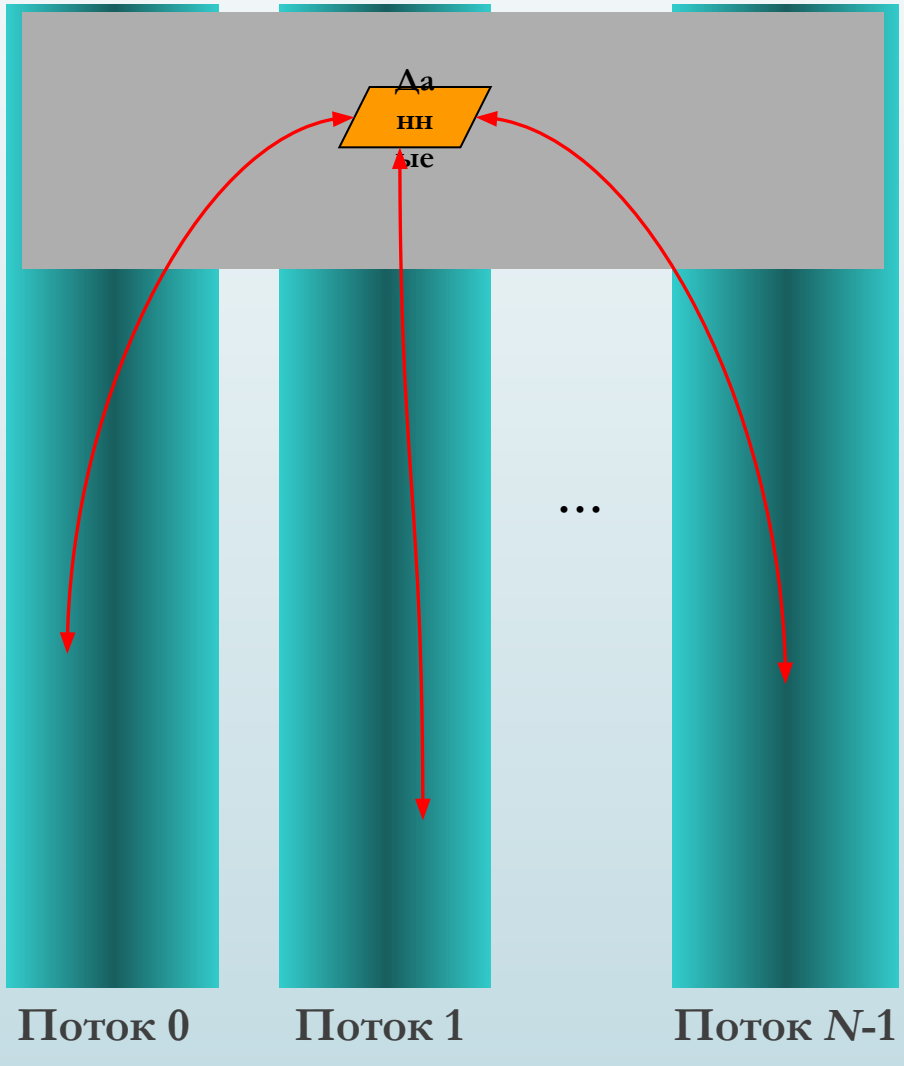


Параллельное программирование


MIMD
Системы с общей оперативной памятью
Системы с распределённой оперативной памятью
Параллельное программирование
программирование в общей памяти
модель передачи сообщений

Программирование в общей памяти

- Параллельное приложение состоит из нескольких потоков выполнения, работающих одновременно
- Потоки разделяют общую память
- Обмен информацией между потоками осуществляются посредством чтения/записи данных в общей памяти
- Потоки могут выполняться как на одном, так и на разных процессорах



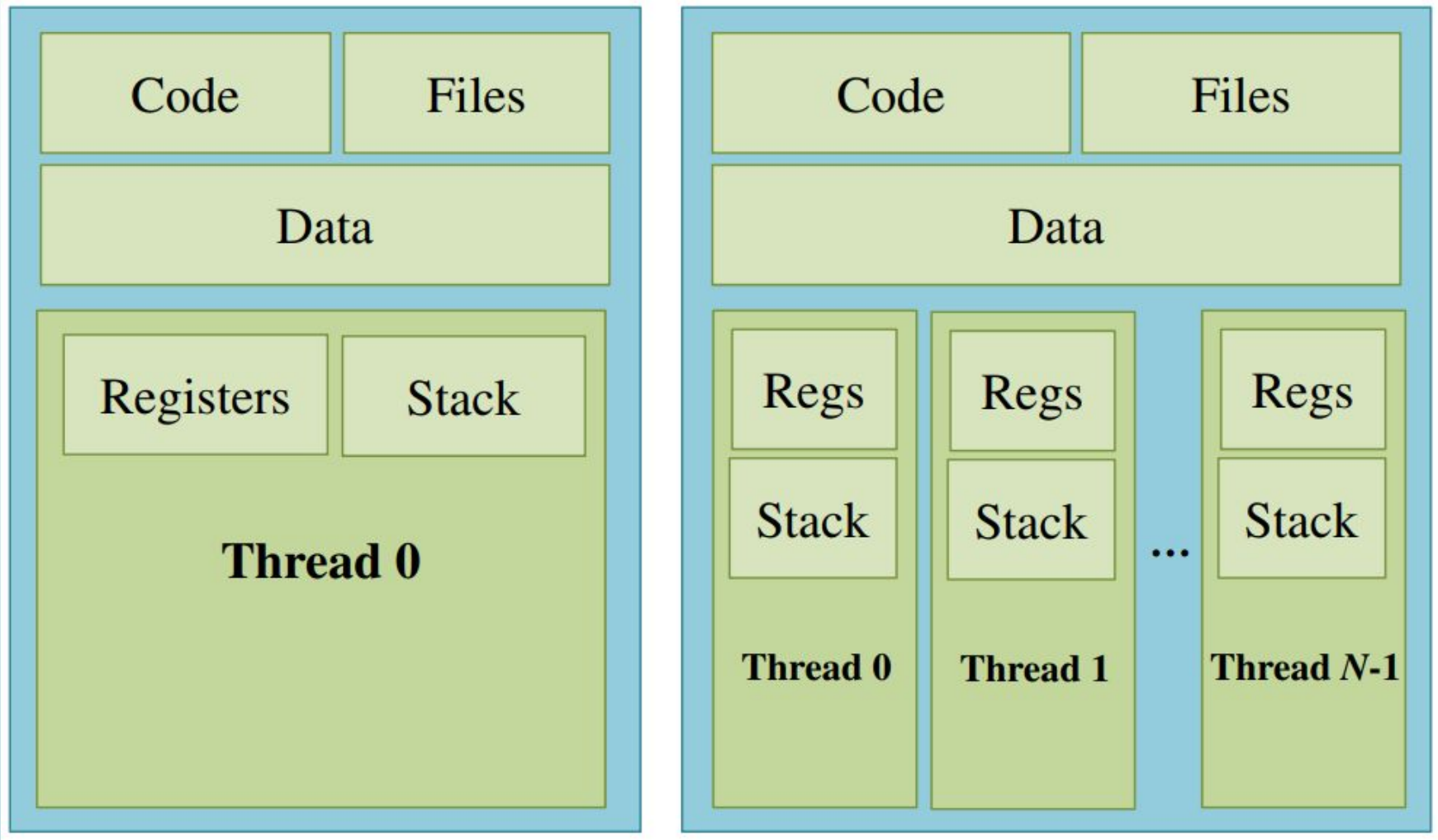
Польза

- Упрощение программы за счёт вынесения механизмов чередования выполнения различных слабо взаимосвязанных подзадач, требующих одновременного выполнения, в отдельную подсистему многопоточности
- Повышение производительности процесса за счёт распараллеливания процессорных вычислений и операций ввода-вывода

Реализация модели с общей памятью

- Процесс (process) – это выполняющийся экземпляр программы, владеющий системными ресурсами (памятью, открытыми файлами, атрибутами безопасности)
- Процесс состоит из одного или нескольких потоков выполнения (threads)
- Поток выполнения (execution thread) – это исполняемый код, который имеет собственный стек и часть контекста процесса (регистры)
- Потоки разделяют ресурсы процесса (код, память – heap, дескрипторы)
- Управляет выполнением потоков планировщик ОС
- Создание и переключение потоков выполняется быстрее, чем аналогичные операции с процессами

Реализация модели с общей памятью



Однопоточное приложение

Многопоточное приложение

Проблемы

- Необходимость синхронизации доступа к общим данным
- Гонки потоков (Data Race) – конкуренция за обращение к данным => необходимость синхронизации обращений (мьютексы, семафоры, критические секции, …) =>
- Взаимные блокировки (Deadlock)

- «плавающие» ошибки, проявляющиеся в случайные моменты времени и пропадающие при попытках их локализовать

Примеры реализаций

- Win32 API Threads

```
HANDLE WINAPI CreateThread( LPSECURITY_ATTRIBUTES lpThreadAttributes,
                            SIZE_T dwStackSize, LPTHREAD_START_ROUTINE lpStartAddress,
                            LPVOID lpParameter, DWORD dwCreationFlags, LPDWORD lpThreadId );
```

- C run-time

```
library uintptr_t _beginthread(void *start_address(void *), unsigned int stack_size, void *arglist);
```

- <threads.h>

```
rc = thrd_create(&tid, threadfun, NULL);
thrd_join(tid, NULL);
```

- Intel Threading Building Blocks

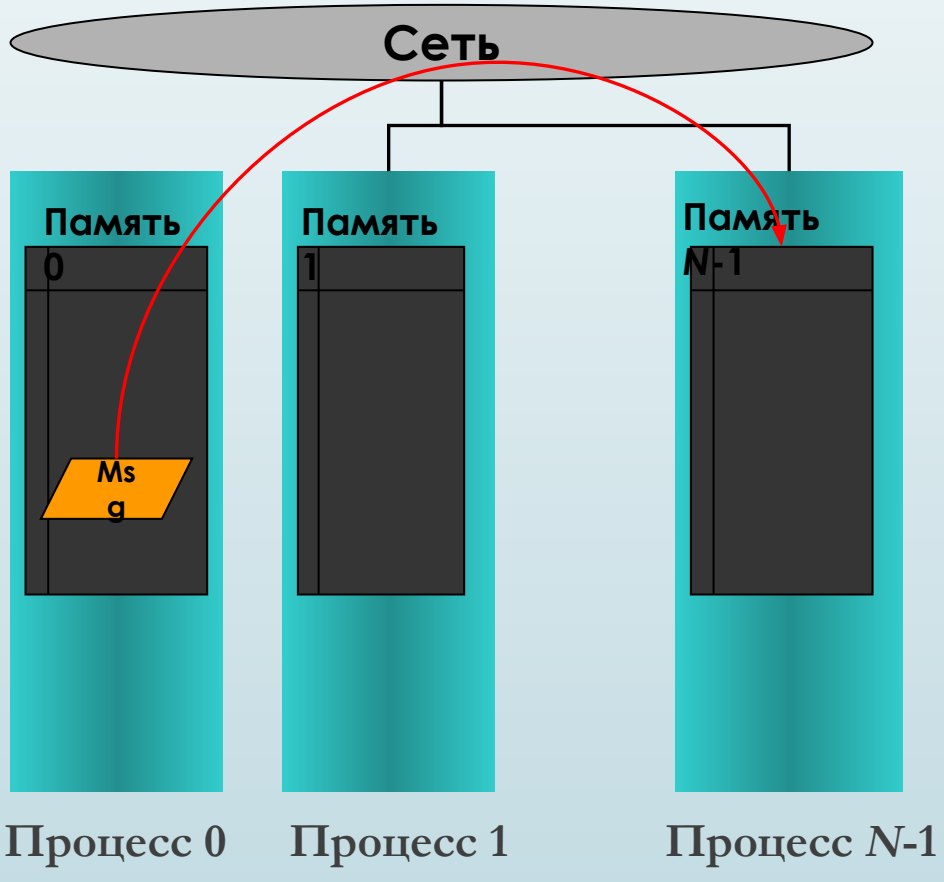
```
task_group tg;
tg.run(MyTask("1"));
tg.run(MyTask("2"));
tg.wait();
```

- OpenMP

```
#pragma omp parallel printf(“Hello!\n”);
```

Модель передачи сообщений

- Параллельное приложение состоит из нескольких процессов, выполняющихся одновременно
- Каждый процесс работает в своем адресном пространстве, каких-либо общих данных нет
- Обмены данными между процессами осуществляются посредством явной отправки/получения сообщений
- Процессы могут выполняться как на одном и том же, так и на разных процессорах



Польза

- Независимость от архитектуры
- Отсутствие проблем с синхронизацией

Реализация модели передачи сообщений

- Процессы могут связываться только через посредника – операционную систему
- При необходимости в обмене данными поток обращается с запросом к ОС
- ОС, пользуясь своими привилегиями, создает различные системные средства связи
- Многие из средств межпроцессного обмена данными выполняют также и функции синхронизации: в том случае, когда данные для процесса-получателя отсутствуют, последний переводится в состояние ожидания средствами ОС, а при поступлении данных от процесса-отправителя процесс-получатель активизируется

Реализация модели передачи сообщений

Передача данных через ОС может осуществляться несколькими способами:

- канал (pipe, конвейер) — псевдофайл, в который один процесс пишет, а другой читает
- сокеты — поддерживаемый ядром механизм, скрывающий особенности среды и позволяющий единообразно взаимодействовать процессам, как на одном компьютере, так и в сети
- почтовые ящики (только в Windows), однонаправленные, возможность широковещательной рассылки
- вызов удаленной процедуры, процесс **A** Может вызвать процедуру в процессе **B**, и получить обратно данные

Реализация модели передачи сообщений

Стандарт MPI (Message Passing Interface)

```
if (rank == 0)
        MPI_Send(&number, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
else
        MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

Гибридные модели?

- Передача сообщений в системе с общей памятью
- DSM (Разделяемая распределенная память)