ГИБРИДНОЕ ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ В СТАНДАРТАХ OpenMP, MPI

Нет загадочнее гибрида жизни для человечества, чем сам человек.

Л.С. Сухоруков

ФПК "Параллельные вычислительные технологии"

Сравнение стандартов

MPI

- Плюсы
  - Переносимость для систем с общей и распределенной памятью
  - Масштабируемость при увеличении узлов
  - Отсутствие проблемы размещения данных
- Минусы
  - Сложность разработки и отладки
  - Высокая латентность, низкая пропускная способность
  - Явные коммуникации
  - Сложность балансировки загрузки

OpenMP

- Плюсы
  - Простая реализация параллелизма
  - Низкая латентность, высокая пропускная способность
  - Неявные коммуникации
  - Динамическая балансировка загрузки
- Минусы
  - Переносимость только для систем с общей памятью
  - Масштабируемость в пределах одного узла
  - Возможная проблема размещения данных
  - Отсутствие возможности задать порядок нитей

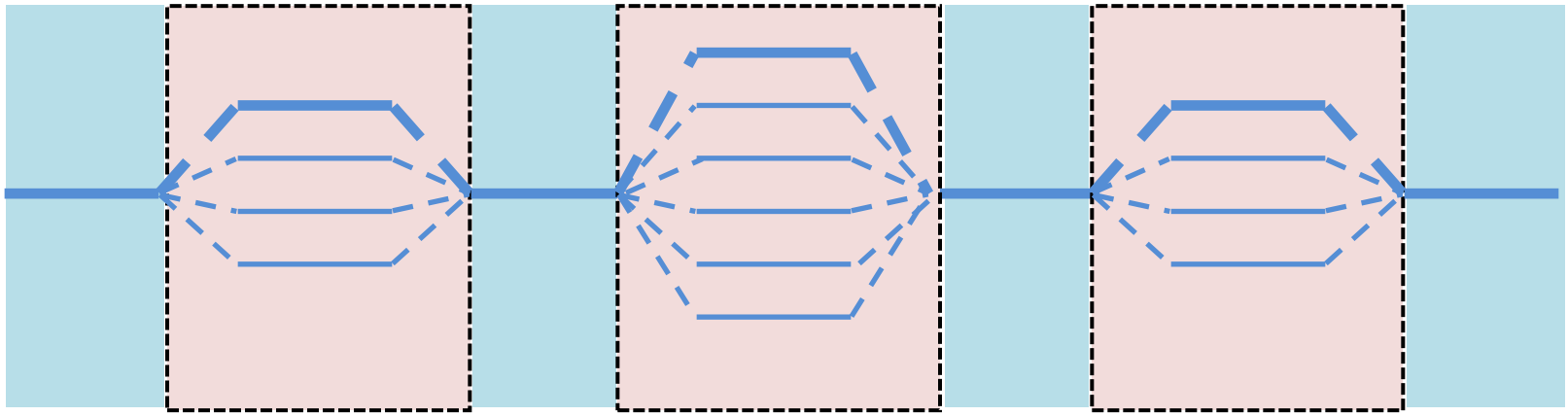
Архитектура MPI+OpenMP: плюсы

- Удобное применение для кластеров с SMP-узлами:
  - MPI – между узлами
    - Избегаем накладных расходов на MPI-коммуникации внутри узла
  - OpenMP – внутри узла
    - Получаем передачу сообщений большего размера за меньшее время и динамическую балансировку загрузки.
- Совпадение с естественной логикой 2-уровневого параллелизма в некоторых задачах.
- Потенциальная возможность получить большее ускорение, чем "чистый" MPI или "чистый" OpenMP.

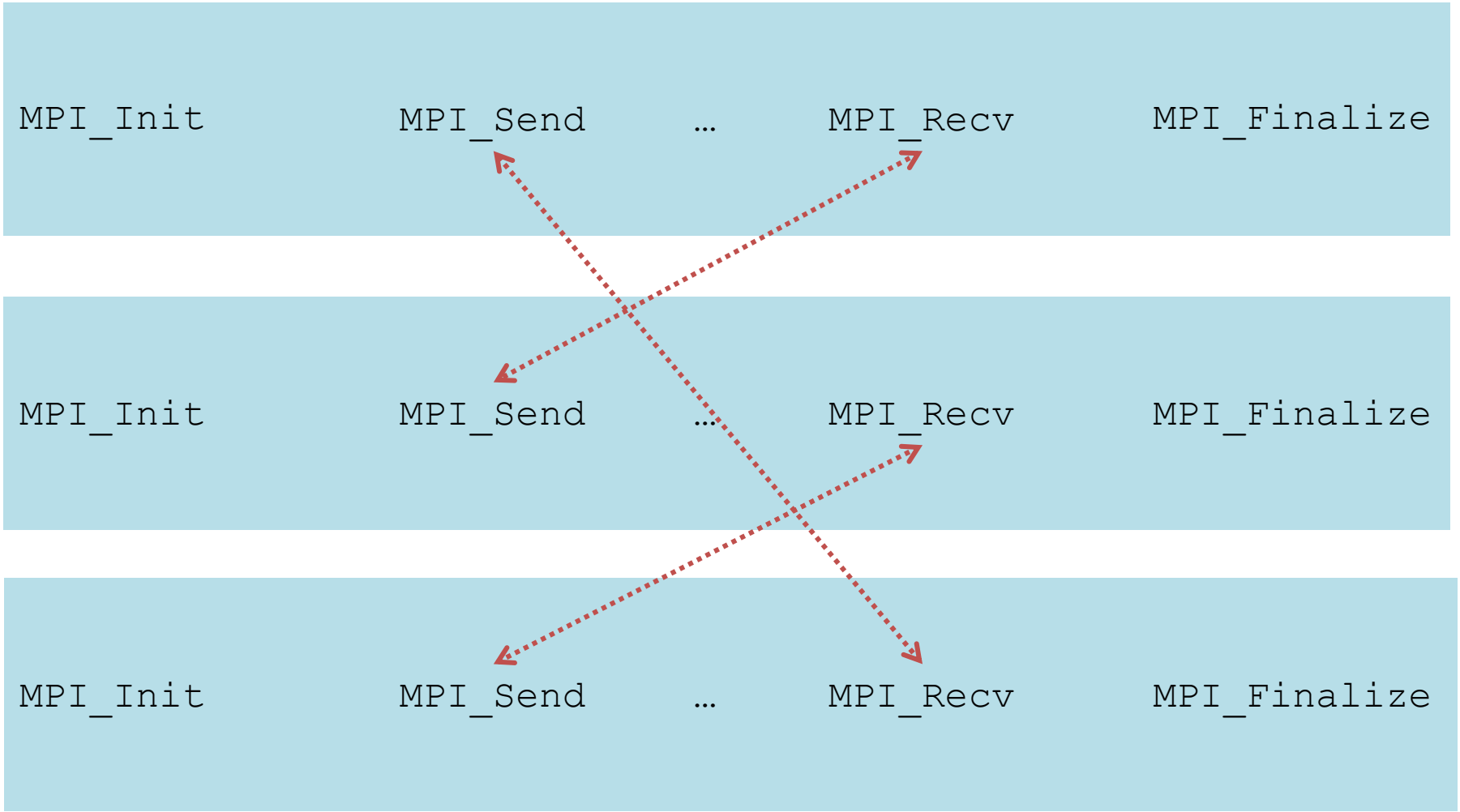
Архитектура MPI+OpenMP: минусы

- Меньшая масштабируемость OpenMP.
- Возможность тупиков в MPI.
- Накладные расходы на обработку нитей:
  - создание нитей
  - когерентность кэшей и размещение данных
  - во время MPI-обмена все нити, кроме одной, бездействуют
    - необходимость пересечения вычислений и коммуникаций для лучшей производительности
    - критическая секция для разделяемых переменных.
- Несовпадение с естественной логикой 1-уровневого параллелизма в некоторых задачах.

OpenMP-программа



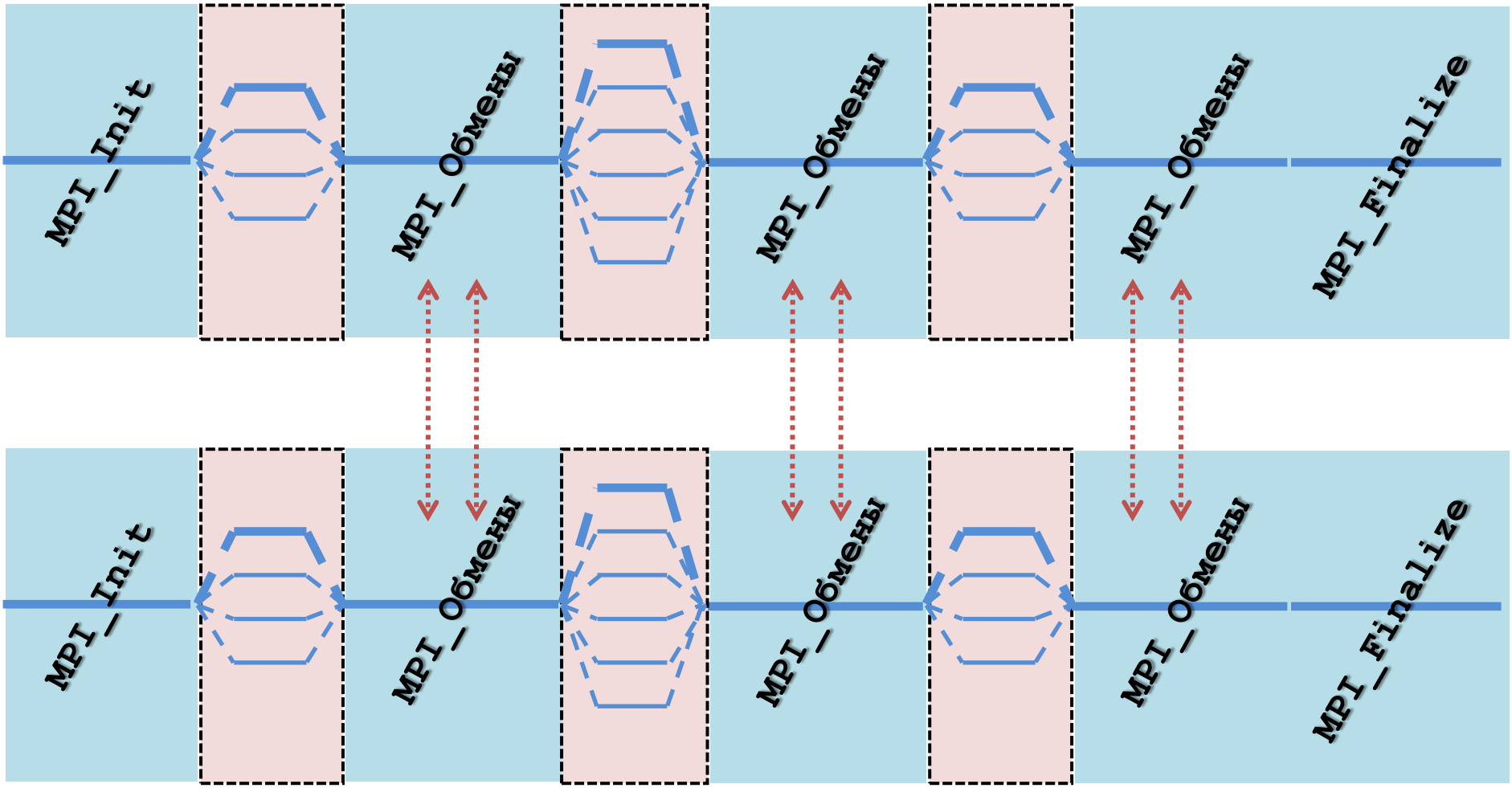
MPI-программа



Подходы к распараллеливанию

- От последовательной программы:
  - разделить на MPI-процессы
  - добавить OpenMP-нити.
- От OpenMP-программы
  - трактовать как последовательный код, добавить MPI-коммуникации
- От MPI-программы
  - добавить OpenMP-нити.
- Простой способ избежать ошибок
  - использовать MPI-коммуникации вне параллельных регионов
  - только нить-мастер может вызывать функции MPI-обменов.

MPI+OpenMP программа



MPI+OpenMP программа


```
#include "mpi.h"
#include "omp.h"
int main(int argc, char *argv[])
{
  int rank, numtasks;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  ... /* Вычисления, вызовы функций MPI-обменов */

  omp_set_num_threads(необходимое_количество_нитей);
  #pragma omp parallel for private(…) shared(…) …
  for (…) {
     …
  }

  ... /* Вычисления, вызовы функций MPI-обменов */
  MPI_Finalize();
  return 0;
}
```

Уровни поддержки нитей в MPI

- MPI_THREAD_SINGLE
  - MPI-процесс исполняет единственную нить
- **MPI_THREAD_FUNNELED**
  - MPI-процесс может запустить несколько нитей
  - MPI-вызовы разрешены только в той нити, которая выполнила инициализацию MPI
- MPI_THREAD_SERIALIZED
  - MPI-процесс может запустить несколько нитей
  - MPI-вызовы разрешены в любой нити, но не более чем одной нитью одновременно
- MPI_THREAD_MULTIPLE
  - MPI-процесс может запустить несколько нитей
  - MPI-вызовы разрешены в любой нити без ограничения на количество одновременных вызовов

MPI-программа с поддержкой нитей


```
#include "mpi.h"
#include <stdio.h>
int main(int argc, char *argv[])
{
  int required = MPI_THREAD_FUNNELED;
  int rank, provided;

  MPI_Init_thread(&argc, &argv, required, &provided);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if (mpi_rank==0) {
    switch (provided) {
    case MPI_THREAD_SINGLE: /* */ break;
    case MPI_THREAD_FUNNELED: /* */ break;
    case MPI_THREAD_SERIALIZED: /* */ break;
    case MPI_THREAD_MULTIPLE: /* */ break;
    default: /* */;
  }
  MPI_Finalize();
  return 0;
}
```

MPI+OpenMP программа


```
// Для MPI_THREAD_FUNNELED
#pragma omp barrier
#pragma omp master
  MPI_xxx(…)
```

```
// Для MPI_THREAD_SERIALIZED
#pragma omp barrier
#pragma omp single
  MPI_xxx(…)
```

```
// Для MPI_THREAD_FUNNELED
#pragma omp parallel
{
  if (my_thread_number==0)
    MPI_xxx(…); // Обмен
  else
    // Вычисления
}
```

Intel Cluster OpenMP