Проблема масштабируемости в MPI: Организация пересылок – последовательная пересылка данных вместо одновременной.

Подготовил Павел Никишкин
323 группа, СКИ ВМК

Московский государственный университет имени М. В. Ломоносова, факультет ВМК

Проблема

При последовательной пересылке сообщений могут возникнуть следующие проблемы:

Низкая производительность: пересылка блокирует процесс до завершения передачи данных, что может вызвать простои в работе процессоров.
Гонки данных: при параллельном обращающении к общей области памяти могут возникнуть ситуации гонки данных – один процесс пытается получить доступ к данным, к которым уже обращается другой процесс.
Неправильная синхронизация: может привести к непредсказуемому поведению программы и сбоям.
Возможность блокировки (deadlock): последовательная пересылка сообщений может вызвать блокировку – процессы ожидают друг друга для выполнения операции, из-за чего программа оказывается заблокированной, неспособной продолжить выполнение.

Пример – скалярное произведение векторов

При поиске скалярного произведения двух векторов существует возможность разбить выполнение алгоритма на несколько процессов – оба вектора разбиваются на одинаковое число блоков, в каждом блоке вычисляется скалярное произведение, а результат вычисляется как их сумма. При этом пересылку сообщений с локальным результатом между процессами можно организовать разными способами.

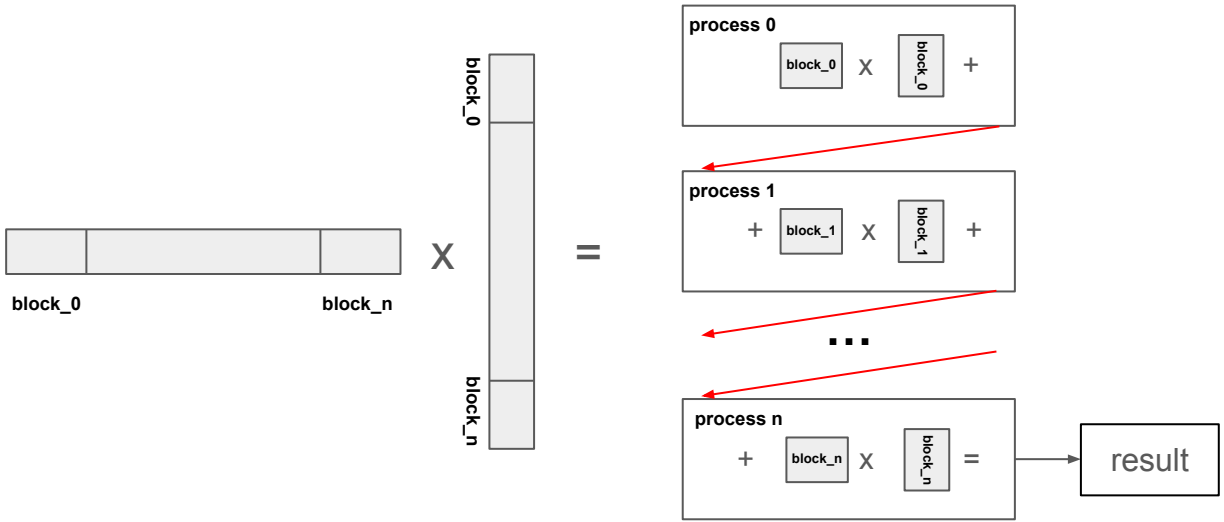

Иллюстрация алгоритма поиска скалярного произведения
с последовательными пересылками

Пример – скалярное произведение векторов

При этом есть возможность одновременной пересылки сообщений и сборки результата в одном процессе:

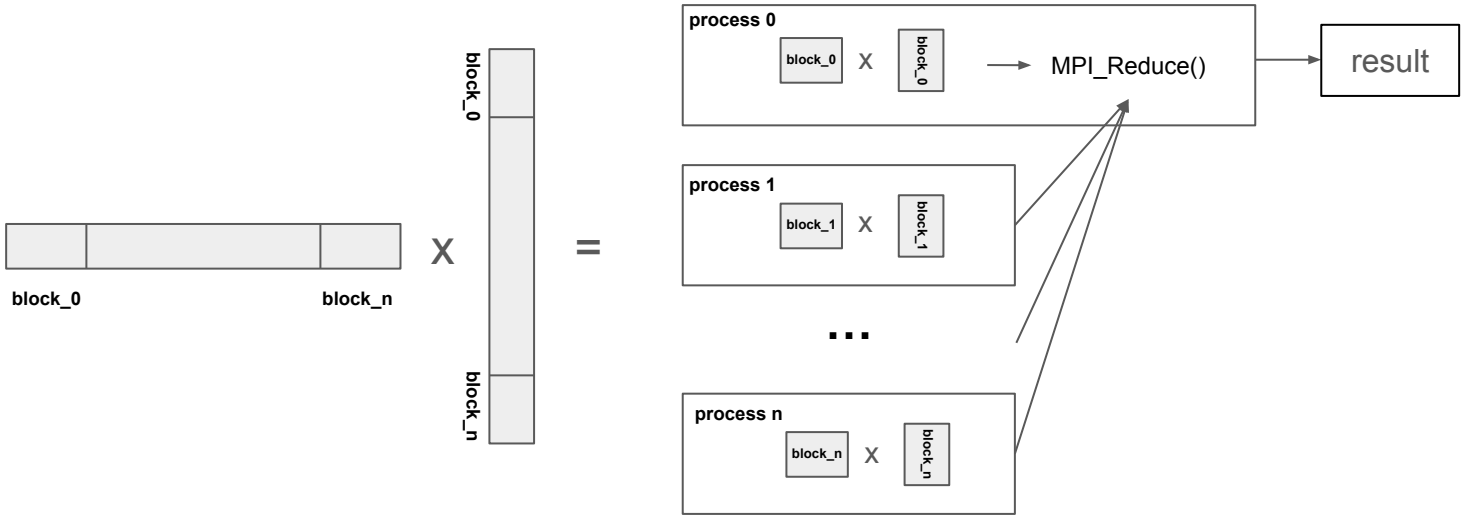

Иллюстрация алгоритма поиска скалярного произведения
с параллельными пересылками

Сравнение версий

Проблемная версия

```
double final_result = 0.0;

// Receive result from previous process
if (rank > 0) {
    MPI_Recv(&final_result, 1, MPI_DOUBLE, rank - 1, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}

// Compute local scalar product
double local_result = scalarProduct(local_vec1, local_vec2, block_size);
final_result += local_result;

// Send local result to next process
if (rank < size - 1) {
    MPI_Send(&final_result, 1, MPI_DOUBLE, rank + 1, 0,
MPI_COMM_WORLD);
}
```

Оптимизированная версия

```
// Compute local scalar product
double localResult = scalarProduct(localVector1, localVector2, n);

// Reduce results to process 0
double globalResult;
MPI_Reduce(&localResult, &globalResult, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);
```

Параметры тестирования

1. 2 массива double из 10^8 элементов.
2. Параметры системы:

   Polus - параллельная вычислительная система, 5 вычислительных узлов.

   Основные характеристики каждого узла:

   - 2 десятиядерных процессора IBM POWER8 (каждое ядро имеет 8 потоков) всего 160 потоков
   - Общая оперативная память 256 Гбайт (в узле 5 оперативная память 1024 Гбайт) с ECC контролем
   - 2 x 1 ТБ 2.5” 7K RPM SATA HDD
   - 2 x NVIDIA Tesla P100 GPU, 16Gb, NVLink
   - 1 порт 100 ГБ/сек

Результаты тестирования

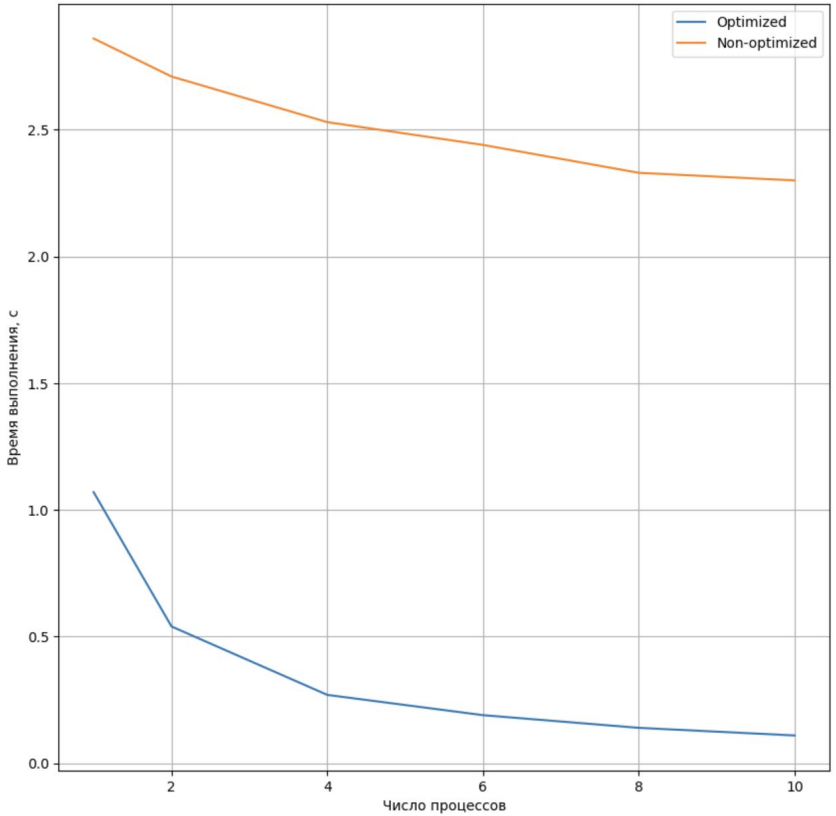

График зависимости времени выполнения от числа процессов

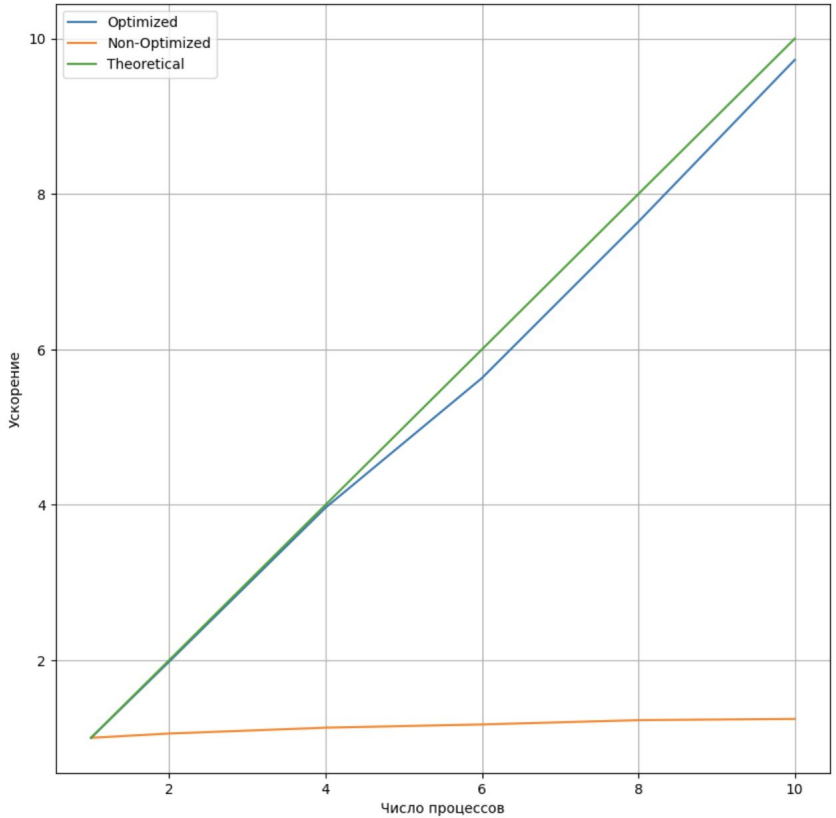

График зависимости ускорения от числа процессов