ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

«Структуры и алгоритмы обработки данных»

на тему

ФИЛЬТР БЛУМА

|  |  |
| --- | --- |
| Выполнил студент | Григорьев Юрий Вадимович |
|  | Ф.И.О. |

|  |  |
| --- | --- |
| Группы | ИС-142 |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работу принял |  | ст. преп. Кафедры ВС Д. М. Берлизов |
|  | подпись |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Защищена |  | Оценка |  |
|  |  |  |  |

Новосибирск – 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#__RefHeading___Toc4327_3153689509)

[1 Алгоритмы коллективных обменов информацией в распределённых ВС 4](#__RefHeading___Toc4329_3153689509)

[1.1 Трансляционный обмен информацией в распределённых ВС 4](#__RefHeading___Toc4331_3153689509)

[1.2 Трансляционно-циклический обмен информацией в распределённых ВС 5](#__RefHeading___Toc4333_3153689509)

[1.3 Выводы 5](#__RefHeading___Toc4335_3153689509)

[2 Экспериментальное исследование эффективности алгоритма 6](#__RefHeading___Toc4337_3153689509)

[2.1 Организация моделирования 6](#__RefHeading___Toc4339_3153689509)

[2.2 Результаты моделирования 6](#__RefHeading___Toc4341_3153689509)

[ЗАКЛЮЧЕНИЕ 7](#__RefHeading___Toc4343_3153689509)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8](#__RefHeading___Toc4345_3153689509)

[ПРИЛОЖЕНИЕ 9](#__RefHeading___Toc4347_3153689509)

[1 Исходный код программы 9](#__RefHeading___Toc4349_3153689509)

ВВЕДЕНИЕ

В распределённых вычислительных системах (ВС) на базе многопроцессорных узлов [1] параллельные алгоритмы и программы разрабатываются на основе стандарта передачи сообщений (message passing). Программы, реализованные в такой модели, состоят из множества процессов (ветвей), которые взаимодействуют посредством сети межмашинных коммуникаций. В стандарте MPI поддерживаются как дифференцированные (point-to-point) обмены, так и коллективные операции, охватывающие множество процессов в рамках одного коммуникатора. Показано [2, 3], что эффективность и масштабируемость параллельных алгоритмов во многом определяется временем работы коллективных операций. Одной из основных и наиболее часто используемых операций стандарта MPI является трансляционный обмен (broadcast), коллективная операция класса «один – всем» (one-to-all). Поддержка операции трансляционного обмена наличествует во всех основных библиотеках, реализующих стандарт MPI: Open MPI, MPICH, MVAPICH, Intel MPI и других.

В данной работе рассматриваются основные алгоритмы трансляционного обмена, реализованные в большинстве программных библиотек стандарта MPI. В модели параллельных вычислений Р. Хокни построены аналитические оценки времени выполнения алгоритмов трансляционного обмена, учитывающих топологические свойства и иерархическую структуру распределённой ВС. Проведены экспериментальные исследования и сравнительный анализ основных алгоритмов трансляционного обмена на пространственно-распределённой кластерной ВС. Рассмотрено применение деревьев принятия решений для решения задачи о выборе оптимального алгоритма трансляционного обмена на основе известного набора входных параметров.

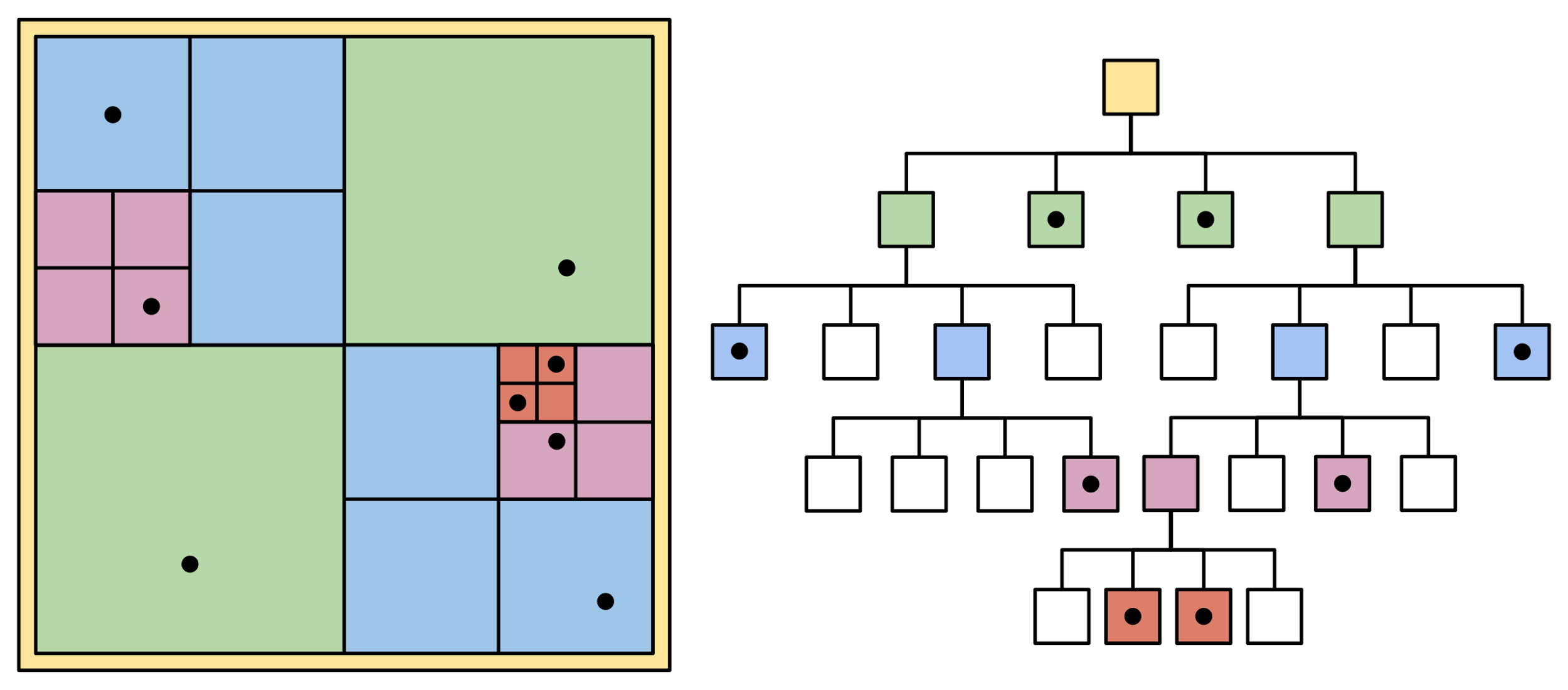
1 Алгоритмы коллективных обменов информацией в распределённых ВС

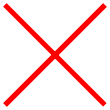
1.1 Трансляционный обмен информацией в распределённых ВС

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Иллюстрации оформляются по центру. Желательно использовать собственноручно сделанные иллюстрации в векторном формате (*SVG*). На каждый рисунок должна иметься ссылка в тексте отчёта.

Деревом квадрантов называется структура данных, предназначенная для рекурсивного разбиения двумерного пространства по четыре области (рис. 1.1).

Рисунок 1.1 – Реализация дерева квадрантов для хранения точек двумерного пространства. На иллюстрации одно и то же дерево представлено как в виде разбиения пространства, так и в виде древовидной структуры

Формулы следует набирать в редакторе формул, например . Если на формулу есть ссылка в тексте, она набирается по центру.

|  |  |
| --- | --- |
|  | (1.1) |

Пример оформления таблиц приведён ниже.

Таблица 1.1 – Основные типы данных, определённые в стандарте MPI

|  |  |
| --- | --- |
| Тип данных MPI | Соответствующий тип данных C |
| MPI\_CHAR | char |
| MPI\_SHORT | signed short int |
| MPI\_INT | signed int |
| MPI\_LONG | signed long int |
| MPI\_UNSIGNED | unsigned int |
| MPI\_FLOAT | float |
| MPI\_DOUBLE | double |

1.2 Трансляционно-циклический обмен информацией в распределённых ВС

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

1.3 Выводы

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

2 Экспериментальное исследование эффективности алгоритма

2.1 Организация моделирования

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

2.2 Результаты моделирования

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы разработан и исследован алгоритм

Осуществлено моделирование разработанного алгоритма. Показано, что

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Хорошевский В. Г.* Распределённые вычислительные системы с программируемой структурой // Вестник СибГУТИ. 2010. № 2 (10). С. 3-41.
2. *Pješivac-Grbović J. et al.* Performance analysis of MPI collective operations // Cluster Computing. – 2007. – Т. 10. – №. 2. – С. 127-143.
3. *Thakur R., Rabenseifner R., Gropp W.* Optimization of collective communication operations in MPICH // The International Journal of High Performance Computing Applications. – 2005. – Т. 19. – №. 1. – С. 49-66.
4. *Хорошевский В. Г.* Архитектура вычислительных систем: Учеб. Пособие – 2‑е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2008. – 520 с.
5. *Legrand A. et al.* Mapping and load-balancing iterative computations // IEEE Transactions on Parallel and Distributed Systems. – 2004. – Т. 15. – №. 6. – С. 546-558.
6. *Евреинов Э. В., Хорошевский В. Г., Пашкеев С. Д.* Однородные вычислительные системы. – Наука, Сиб. Отд-ние, 1978.
7. *Rabenseifner R.* Automatic MPI counter profiling // 42nd CUG conference. – 2000. – С. 396-405.
8. *Bruck J. et al.* Efficient algorithms for all-to-all communications in multiport message-passing systems // IEEE Transactions on parallel and distributed systems. – 1997. – Т. 8. – №. 11. – С. 1143-1156.

ПРИЛОЖЕНИЕ

1 Исходный код программы

/\*

\* A binomial tree broadcast algorithm. Good for short messages sent

\* among a small number of processes.

\*/

**void** Bcast\_binomial(**void** \*data, **int** count, **MPI\_Datatype** datatype,

**int** root, **MPI\_Comm** communicator)

{

**int** rank, comm\_size, src, dst;

**int** relative\_rank, mask;

**int** nbytes = 0;

**int** recvd\_size;

**int** type\_size;

**MPI\_Status** status;

MPI\_Comm\_size(communicator, &comm\_size);

MPI\_Comm\_rank(communicator, &rank);

**if** (comm\_size == 1)

**return**;

MPI\_Type\_size(datatype, &type\_size);

nbytes = type\_size \* count;

**if** (nbytes == 0)

**return**;

relative\_rank = (rank >= root) ? rank - root : rank - root +

comm\_size;

mask = 0x1;

**while** (mask < comm\_size) {

**if** (relative\_rank & mask) {

src = rank - mask;

**if** (src < 0)

src += comm\_size;

MPI\_Recv(data, count, datatype, src, 0, communicator,

&status);

MPI\_Get\_count(&status, MPI\_BYTE, &recvd\_size);

**break**;

}

mask <<= 1;

}

mask >>= 1;

**while** (mask > 0) {

**if** (relative\_rank + mask < comm\_size) {

dst = rank + mask;

**if** (dst >= comm\_size)

dst -= comm\_size;

MPI\_Send(data, count, datatype, dst, 0, communicator);

}

mask >>= 1;

}

}