Тестовый стенд для анализа распределенных алгоритмов балансировки нагрузки

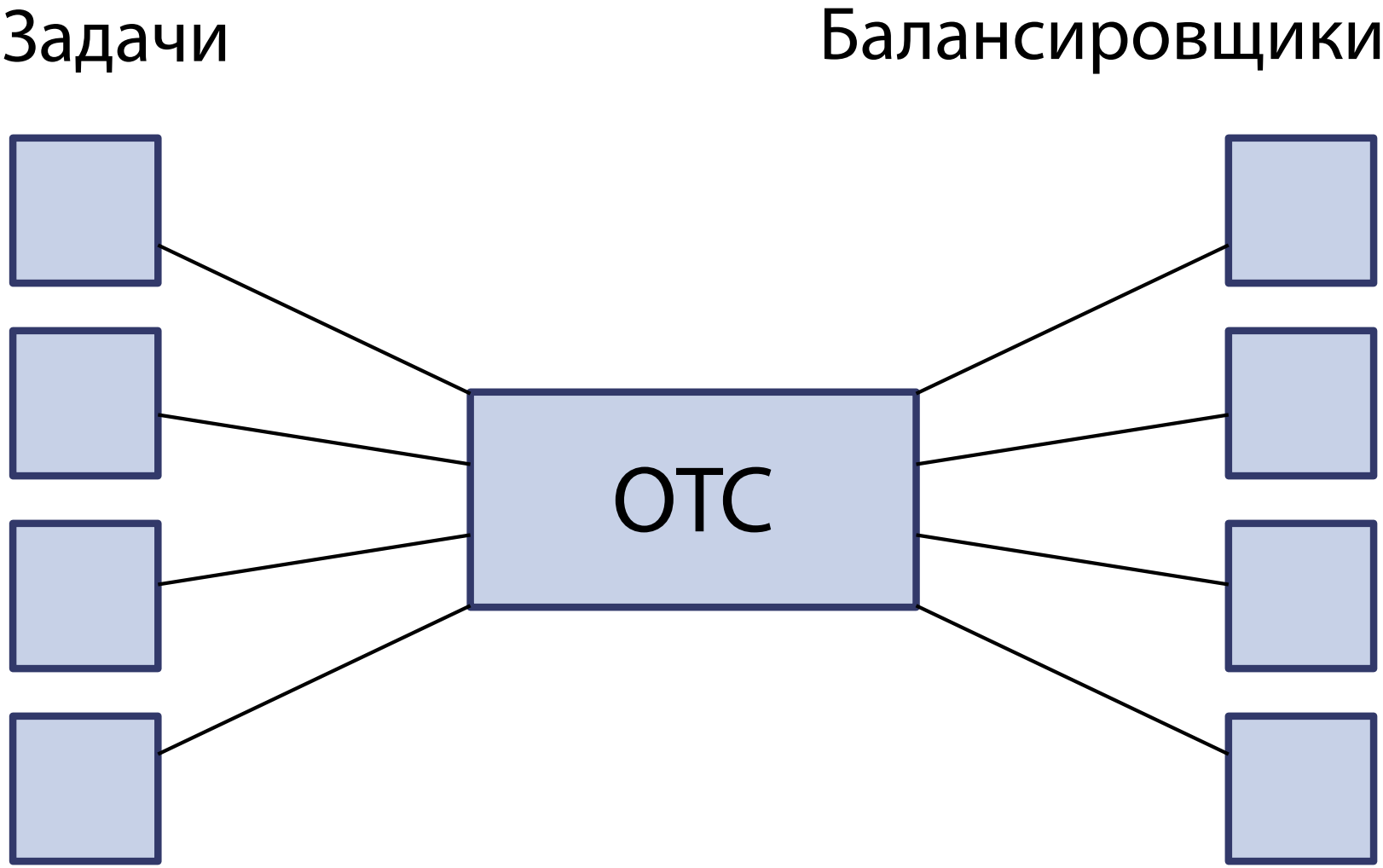
Перепелкин В. А. [1], Сумбатянц И. И. [2]

[2] Новосибирский Государственный Университет
[1] ИВМиМГ СО РАН
Лаборатория Синтеза параллельных программ

Проблема

- Необходимость в статической или динамической балансировке нагрузки при параллельной реализации итерационных численных методов на сетках
- Трудность аналитической оценки распределенных алгоритмов динамической балансировки
- Разностороннее тестирование алгоритма балансировки на разных задачах и конфигурациях алгоритма

Тестовый стенд



Постановка задачи

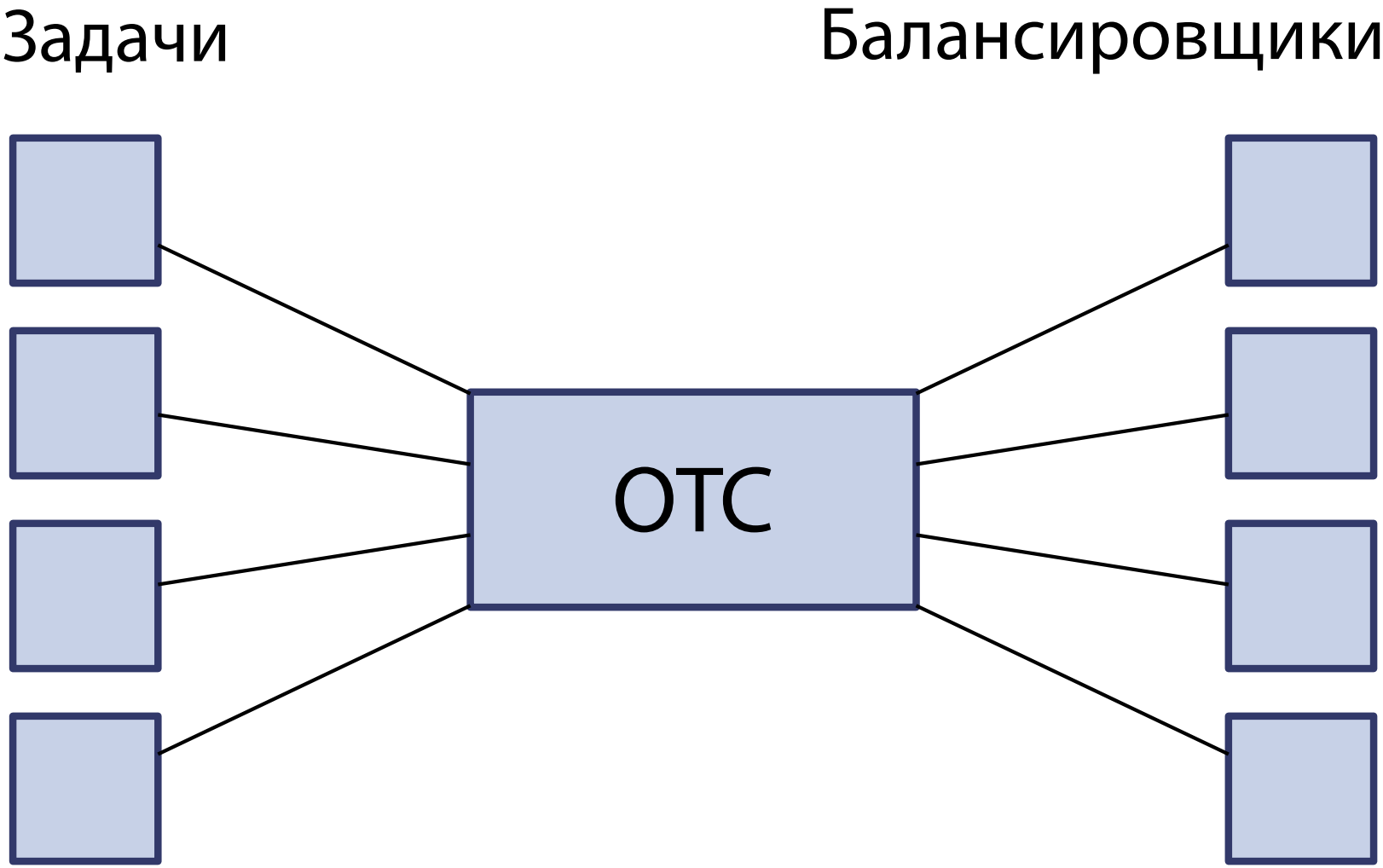
Необходимо разработать и реализовать тестовый стенд, который будет соответствовать следующим требованиям:

- Поддержка распределенных реализаций итерационных сеточных процессов
- Поддержка распределенных локальных динамических алгоритмов балансировки
- Возможность независимо заменять реализации алгоритма балансировки и итерационного сеточного процесса
- Возможность исполнения задачи на мультикомпьютере с виртуальной топологией, постоенной на основе топологии вычислителя
- Возможность определять пользователю конфигурацию логического вычислителя
- Вход: реализации задачи и балансировщика
- Выход: данные, собранные в процессе исполнения задачи

Данные, представляющие интерес для тестирования

- Общее время исполнения и время моделирования
- Развертка максимума и минимума нагрузки по узлам по времени
- Максимальное количество нагрузки, отображенной на узел
- Развертка нагрузки на вычислители по времени
- Общее количество балансировок
- Развертка нагрузки на коммуникационную сеть по времени

Тестовый стенд



Модель балансировщика

- Балансировщик – реализация распределенного алгоритма динамической балансировки и предикат, по которому можно определить необходимость в инициации балансировки
- Входные данные:
  - Нагрузка в окрестности узла и на сам узел
  - Входные параметры, определяемые пользователем

Такая модель может быть использована для реализации диффузионных балансировщиков нагрузки.

Модель задачи

Пусть некоторый класс итерационных численных методов на сетках можно представить в виде кортежа:
$< \boldsymbol{F}, \boldsymbol{N}, \boldsymbol{op}, \boldsymbol{R} >$, где:

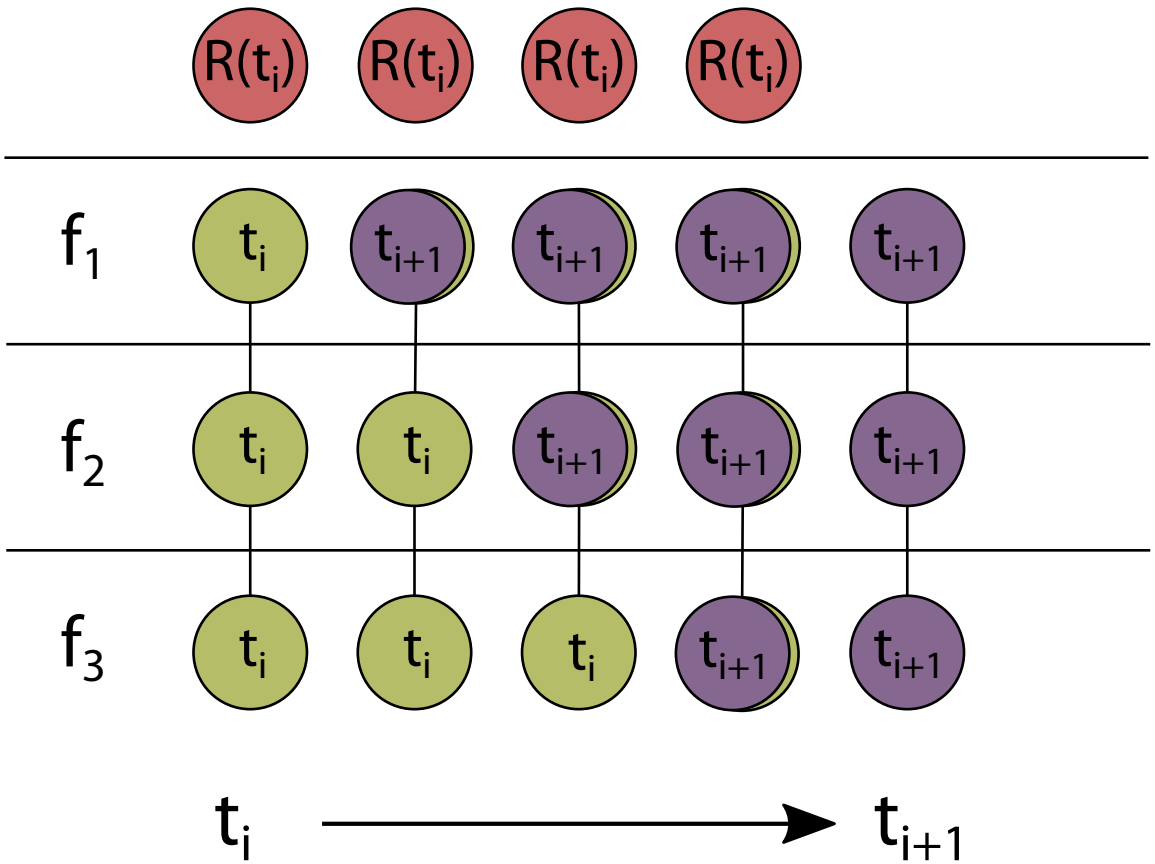
- $\boldsymbol{F}$ – множество вершин графа (фрагментов)
- $\boldsymbol{N}$ – множество ребер на $\boldsymbol{F}$ (отношение соседства)
- $\boldsymbol{op}$ – оператор шага итерации
- $\boldsymbol{R}$ – оператор редукции

Означивание фрагментов

- Процесс исполнения происходит в дискретном времени $T = \{t_1 \ldots t_n\}$
- В каждый момент времени t_i элемент $f \in F$ имеет некоторое означивание (состояние) f_{t_i}
- Операторы шага итерации и редукции применяются к означиванию фрагментов.

Вычисление означивания фрагментов

$$\forall f \in F(f_{t_{i+1}} = op(f_{t_i}, \{fn_{t_i} : (fn \in F) \wedge ((fn, f) \in N)\}, R(t_i))$$



Реализация модели задачи на мультикомпьютере

- Параллельная реализация такой модели – разбиение множества фрагментов и последующее отображение подмножеств на узлы мультикомпьютера
- Разбиение множества ***F*** может быть построено статически или динамически
- Изменение разбиения в процессе исполнения будем называть ***балансировкой***

В такую модель вычислений укладываются метод частиц-в-ячейках, явные разностные схемы и многие клеточные автоматы.

Реализация ОТС

ОТС (Отладочный тестовый стенд) состоит из:

- Системы исполнения реализации задачи
- Интерфейс для подключения задачи и балансировщика

Вход ОТС:

- Описание задачи
- Балансировщик

Интерфейс для описания задачи

Описание задачи:

- Задача – множество объектов класса «фрагмент»
- Реализация фрагмента – реализация системных интерфейсов (данные и операторы итерационного шага и редукции)
- Отношение соседства описывается при инстанцировании фрагмента
- Управление исполнением задачи производится с помощью установки флагов, которые описаны в системных интерфейсах

ОТС скрывает в себе:

- Применение оператора к фрагменту
- Поиск соседних фрагментов
- Редукцию данных
- Миграцию фрагмента

Управление исполнением

Доступные флаги:

- Редукция данных
- Необходимость в данных соседних фрагментов
- Обновление состояния фрагмента для глобального использования
- Переход на следующий шаг итерации

Интерфейс для описания балансировщика

- Реализация алгоритма балансировки с точки зрения ОТС – это процедура, которая может быть вызвана на каждом узле, где запущен ОТС
- ОТС периодически обращается к этой процедуре для определения необходимости инициации балансировки
- Реализация алгоритма балансировки работает в терминах вычислительной нагрузки
  - Вход: нагрузка в окрестности узла
  - Выход: решение по распределению вычислительной нагрузки в окрестности
- ОТС в соответствии с решением балансировщика самостоятельно решает какие именно фрагменты будут переданы

Пример тестирования алгоритма балансировки

- Задача: трассировка лучей
- Декомпозиция: разбиение изображения на слои
- Дисбаланс: некоторые области экрана содержат точки с большим количеством отражений
- Балансировщик: диффузионный балансировщик
- Порог дисбаланса: изменение нагрузки на 1/5
- Нагрузка на узел: количество необработаных пикселей на узле

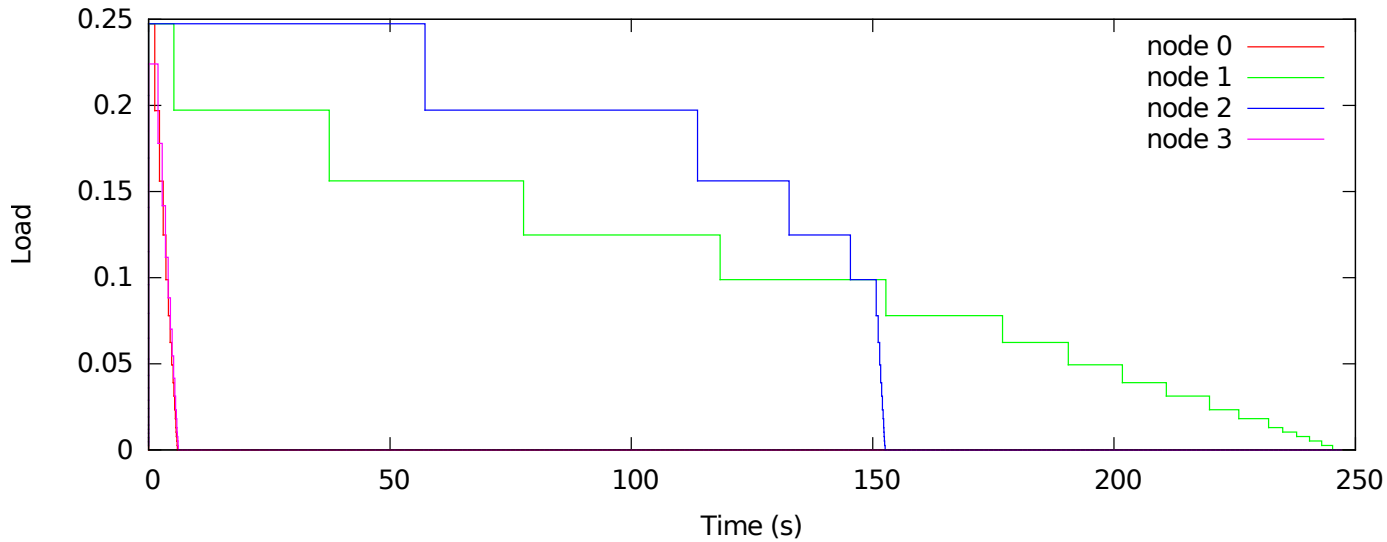
Время исполнения

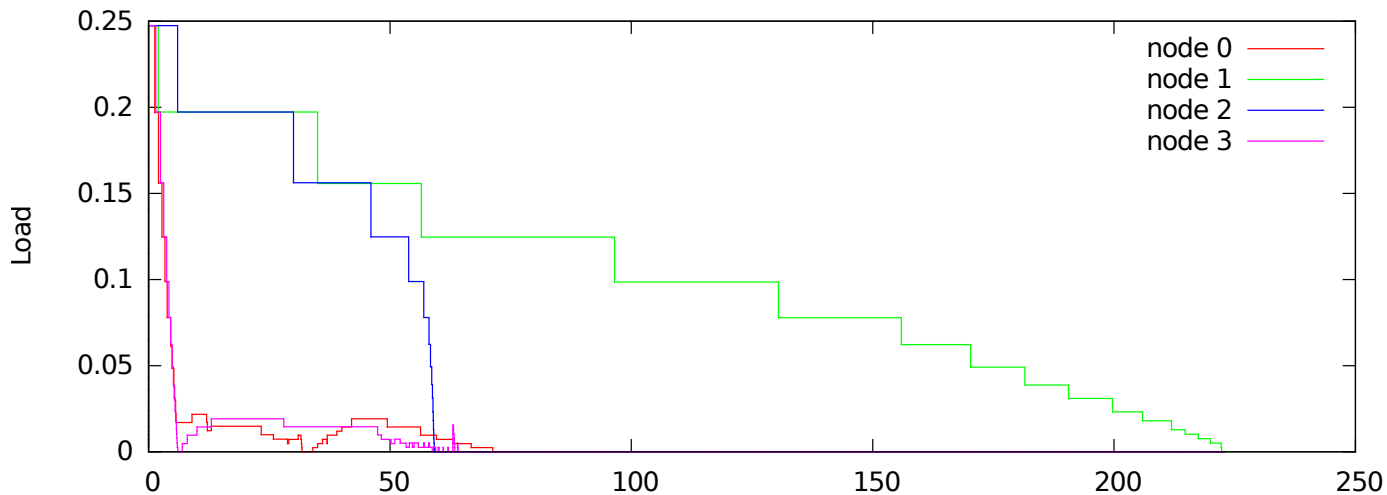
- Изображение 5000x5000 пикселей
- 100 фрагментов

Время исполнения:

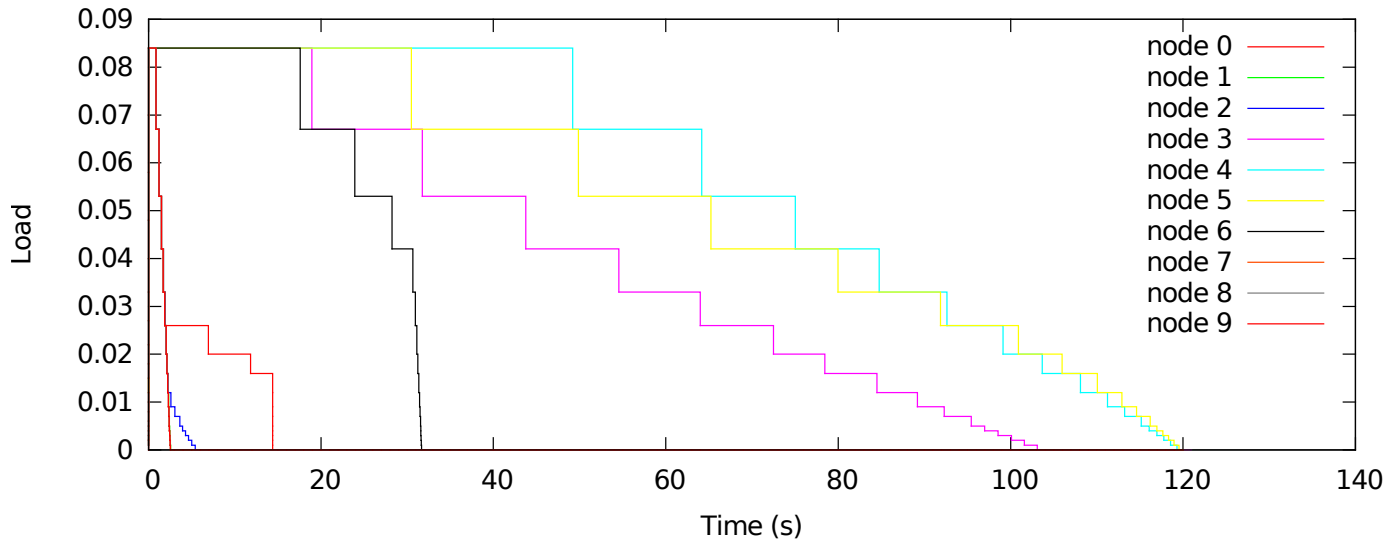
Количество узлов	Без балансировки	С балансировкой
4	243 сек	224 сек
10	119 сек	104 сек

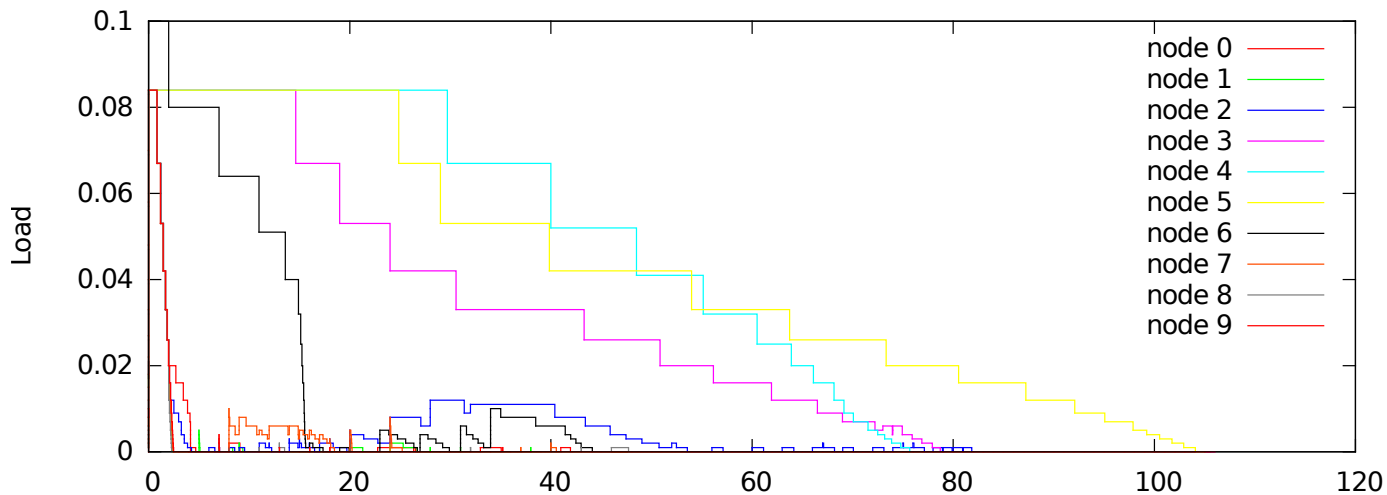
Развертка распределения нагрузки по узлам (4 узла)





Развертка распределения нагрузки по узлам (10 узлов)





Результаты тестирования

Возможные проблемы алгоритма балансировки, выявленные при тестировании:

- Порог дисбаланса
- Функция оценки вычислительной нагрузки

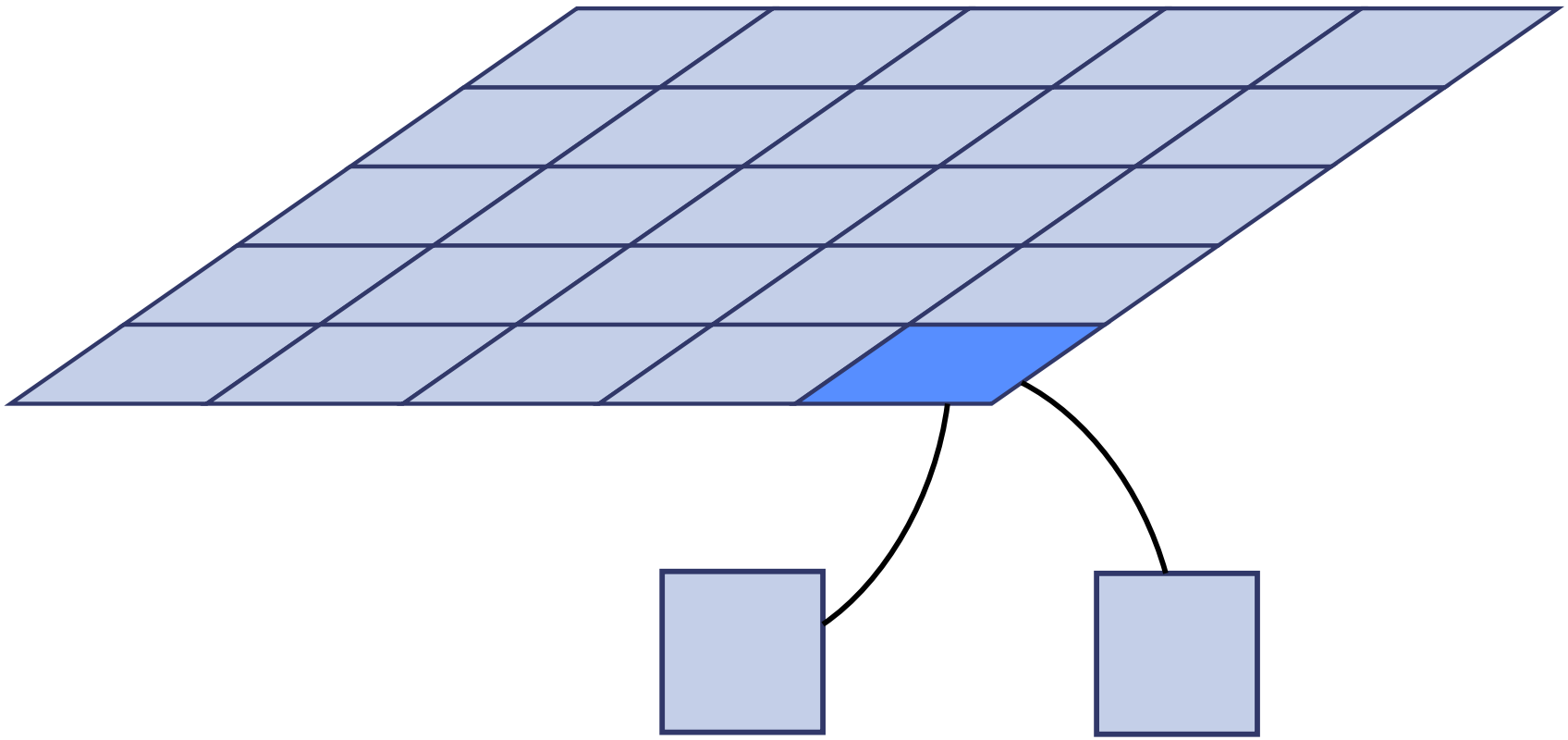
Заключение

- Исследованы методы оценки алгоритмов балансировки для целевого класса задач
- Создан прототип системы тестирования
- Реализована задача трассировки лучей и диффузионный балансировщик
- Произведено тестирование балансировщика для задачи трассировки лучей

Литература:

- "Assembly technology for parallel realization of numerical models on MIMD-multicomputers", M.A. Kraeva, V.E. Malyshkin, 2001
- "Implementation of PIC method on MIMD multicomputers with assembly technology", M.A. Kraeva, V.E. Malyshkin, 1997
- "Dynamic load balancing for a 2D concurrent plasma PIC code", Ferraro R.D., 1993
- "OhHelp: a scalable domain-decomposing dynamic load balancing for particle-in-cell simulations", Nakashima Hiroshi, 2009
- "A parallel 3D particle-in-cell code with dynamic load balancing", Wolfheimer, Weiland, 2006

Виртуальный фрагмент



Архитектура