Оптимальное распределение нагрузки на совокупность разнородных узлов

Хританков А.С. МФТИ

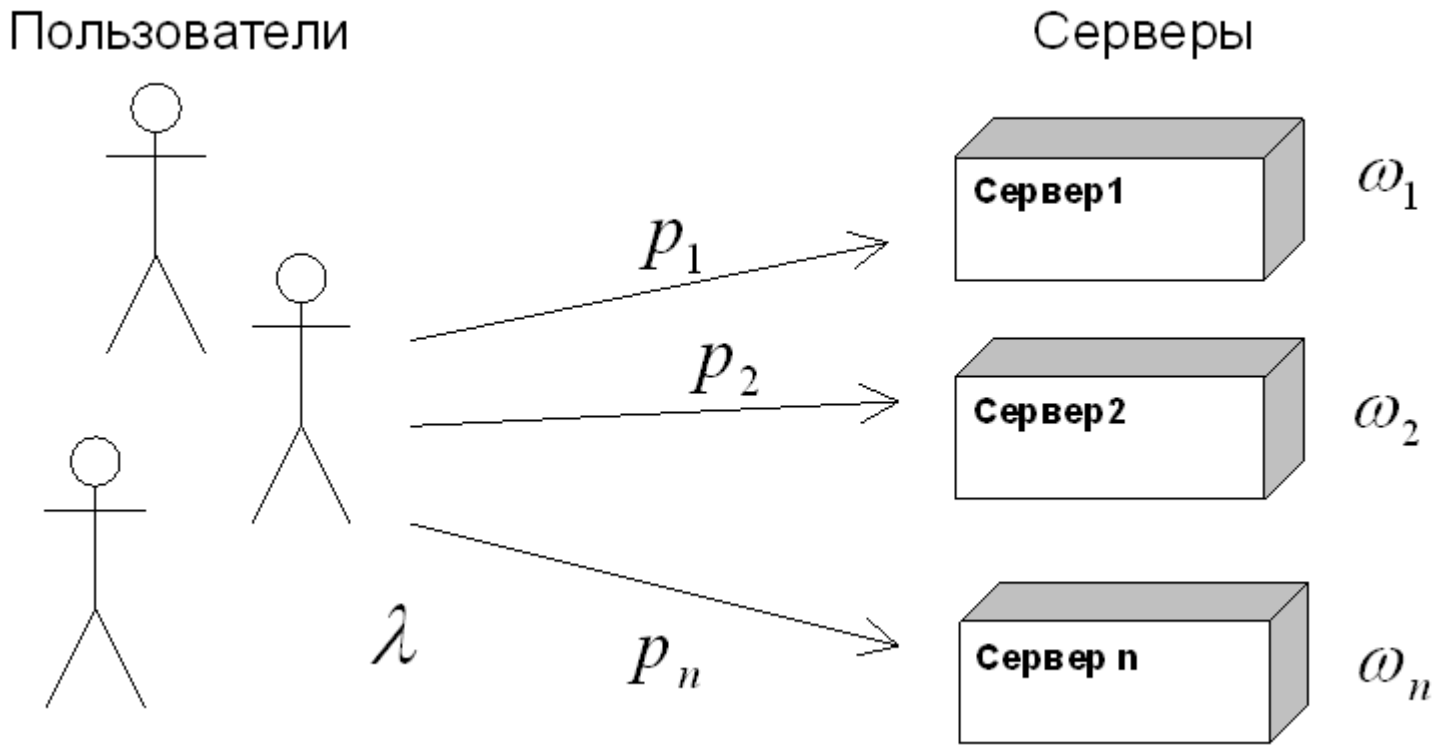
сентябрь 2005

1 международная конференция «Системный анализ и информационные технологии», Переславль-Залесский, 13 сентября 2005

Введение

- Распределенные вычисления и обработка данных
- Необходимость распределения нагрузки
- Системы обмена сообщениями и им подобные, системы запрос-ответ.
- IARNet

Постановка задачи



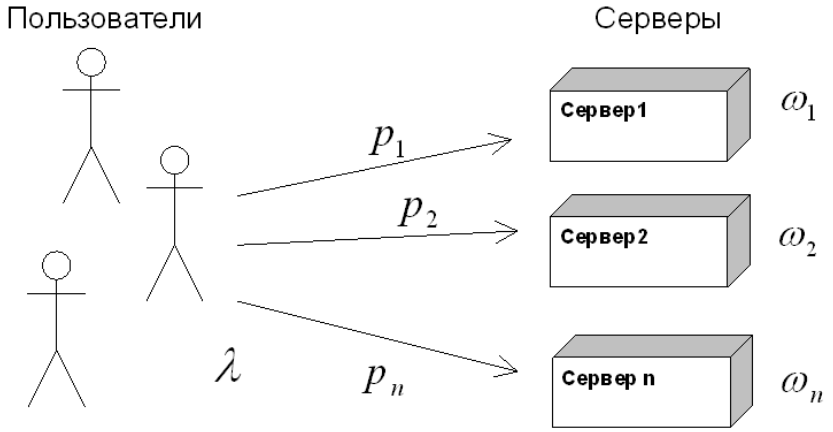
Модель системы

Модель системы

- Система состоит из пользователей и серверов. Пользователи создают запросы, обрабатываемые серверами.
- Запросы считаются *одинаковыми*.
- Система рассматривается в стационарном случае.
- С каждым обработчиком связана очередь ожидания.
- Запросы не теряются.

Постановка задачи.

- Как необходимо распределять создаваемые пользователями запросы по серверам, чтобы *среднее время обслуживания запросов было минимально?*



Существующие подходы

- *Weighted Round Robin(равномерное с весами)*
  - допускает перегрузку серверов при неполной загрузке системы
- *LeastQueue(минимальная очередь)*
  - количество ожидающих запросов на всех серверах одинакова
  - полностью загружает серверы системы
- *LeastLoad(минимальная загрузка сервера)*
  - нагрузка на все серверы одинакова
  - парадокс Браеса – добавление серверов в систему увеличивает время ответа

Проблемы

- Неоптимальность применяемых алгоритмов балансировки
  - Дилемма: оптимальность - производительность
- Ориентация на одинаковые обрабатывающие узлы
  - Проблемы модернизации систем
- Считается, что все запросы примерно одинаковые - однородность нагрузки
- Динамические алгоритмы vs. статические алгоритмы балансировки
  - Высокая требовательность к ресурсам
  - Необходимость отслеживать состояние балансируемой системы

Другие работы

- A. A. Tantawi, D. Towsley (1985)
  - A. A. Tantawi and D. Towsley, ”Optimal Static Load Balancing in distributed Computer Systems,” Journal of the ACM, 32(2), 445-465(1985)
- C. Kim, H. Kameda (1992)
  - Chonggun Kim, Hisao Kameda, “An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems”. IEEE TRANSACTIONS ON COMPUTERS, VOL.41, NO.3, MARCH 1992.
- Рассмотрение статической балансировки сети узлов до начала работы системы. Численные методы нахождения решения.

Постановка задачи

Предположения

- Поведение каждого сервера подчиняется ТМО.
- Все серверы считаются однолинейными(однопоточными)
- Допустимо использование стохастического подхода – решение можно описать набором чисел, выражающих вероятности направления запроса на каждый сервер

Основные обозначения

- ω_i – параметр распределения времени ответа i-го сервера, выражает производительность сервера
- λ – параметр распределения интервала времени между двумя последовательными запросами в потоке, выражает нагрузку системы в целом
- p_i – вероятность направить запрос на i-ый сервер, искомая величина
- δ_i – задержка коммуникации с i-м сервером
- λ^i – точки старта серверов, такое значение нагрузки системы при котором i-ый сервер подключается к обработке запросов пользователей

Построение задачи

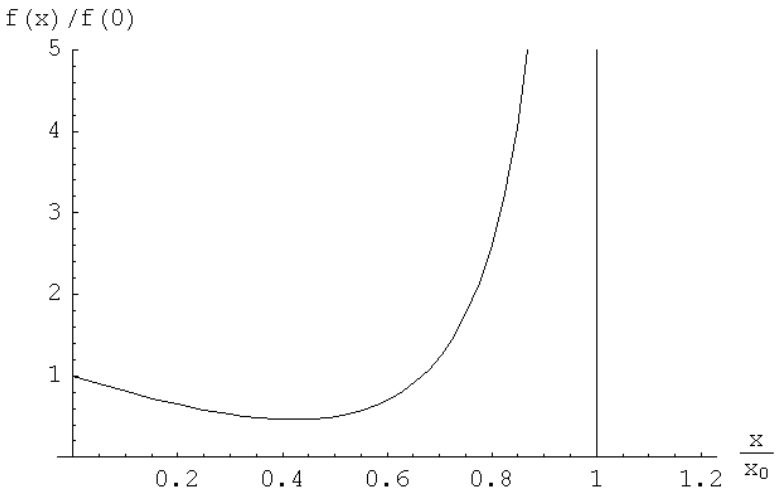
- Из теории массового обслуживания время ответа сервера:

$$T_i = \delta_i + \frac{1}{\omega_i - (\lambda p_i)}$$

- Сервер не должен перегружаться:

$$\omega_i - (\lambda p_i) > 0$$

- Общий вид функции T_i



Математическая формулировка

- Сепарабельная задача оптимизации со строго выпуклым функционалом и линейными ограничениями

$$\vec{p} = \arg\min_{\vec{P}} T = \arg\min_{\vec{P}} \sum_{i=1}^{n} p_i T_i$$

при условии, что

$$\omega_i - (\lambda p_i) > 0,\ p_i \geq 0,\ i = 1..n;\ \sum_{i=1}^{n} p_i = 1;$$

причем $\delta_i \geq 0,\ \omega_i > 0,\ n \in N,\ \lambda \geq 0$ - известны

Этапы решения

- Построение двойственной задачи.
- Решение двойственной задачи
  - эффект старта разных серверов при различной нагрузке на систему – *точки старта серверов* λ^i .
  - техническое упрощение: $\delta_i = \delta = const, \quad \forall i = 1..n$
- Исследование решения двойственной задачи
  - *непрерывная дифференцируемость целевой функции* по λ на оптимальном решении.
- Решение исходной задачи

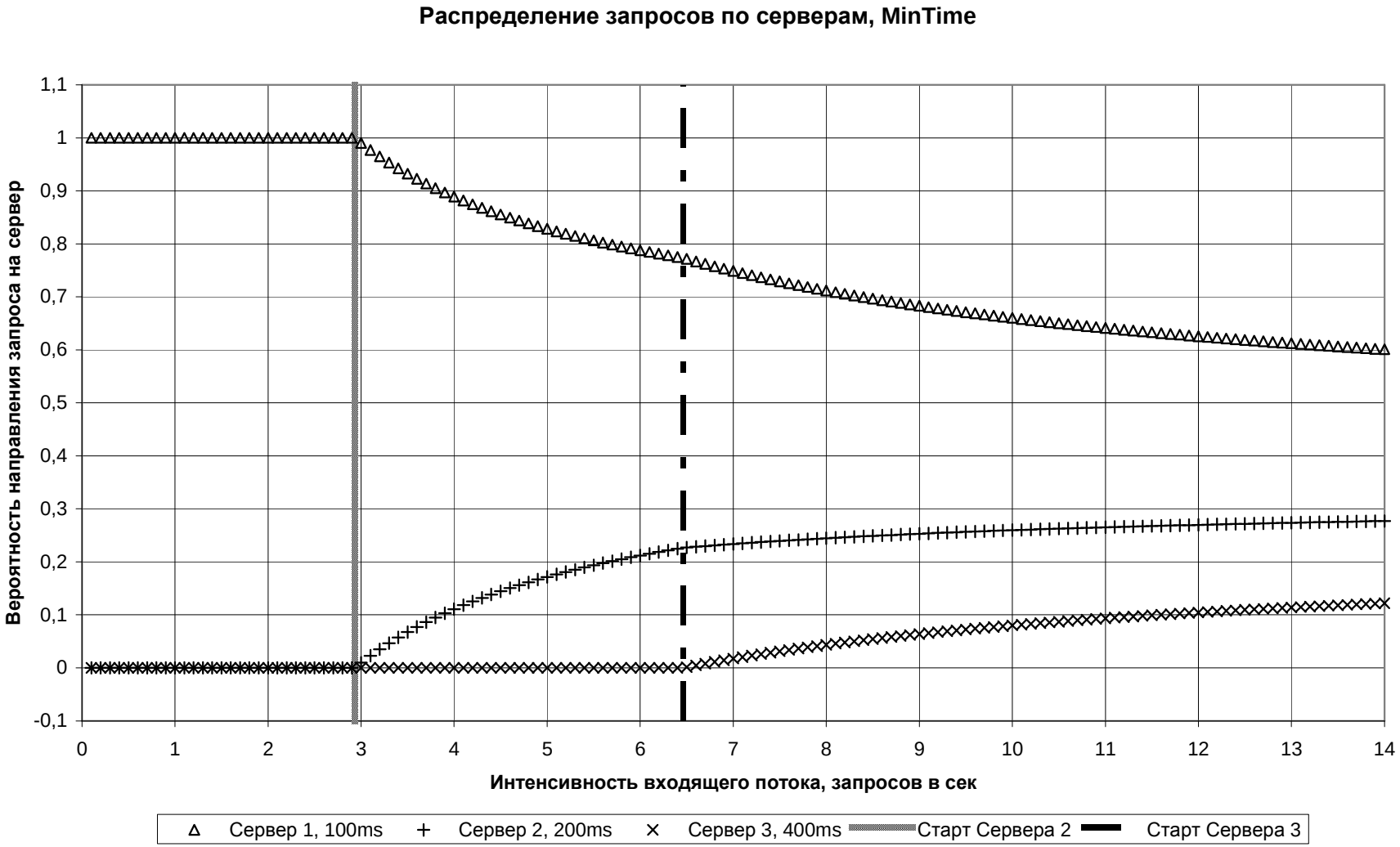
Алгоритм MinTime

1) Расположить серверы по убыванию производительности
2) Найти точку старта для каждого сервера
3) Рассчитать матрицу балансировки R
4) При заданном значении интенсивности входящего потока λ решить задачу минимизации на упорядоченной последовательности
5) Рассчитать вероятность выбора для каждого сервера
6) При поступлении запроса выбрать сервер в соответствии с набором вероятностей

Анализ алгоритма

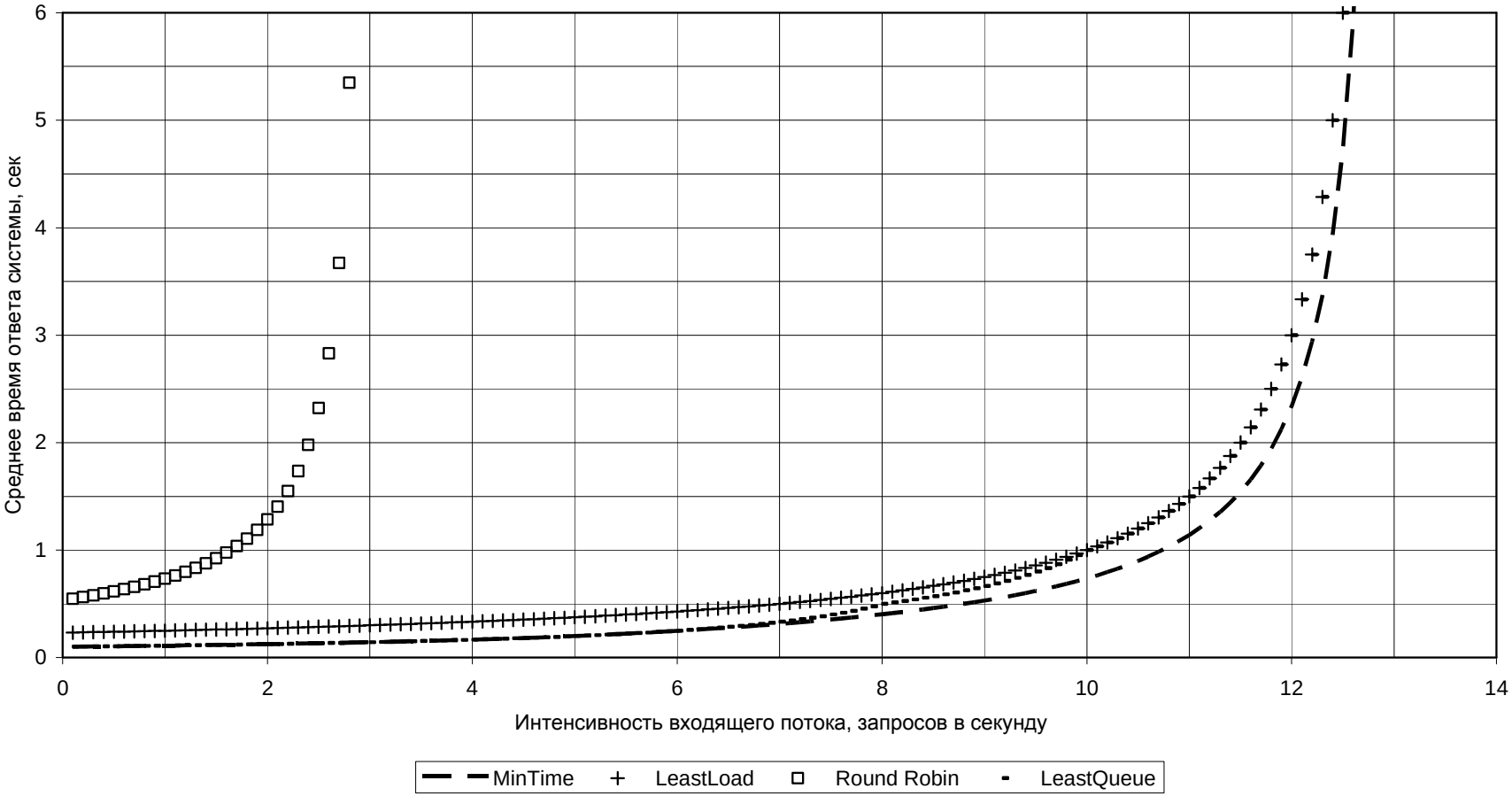
- Шаг 1) требует $O(n \log n)$
- Шаг 2) требует $O(n)$
- Шаг 3) выполним за $O(n^2)$
- Шаг 4) требует $O(\log n)$
- Шаг 5) требует $O(n)$
- Шаг 6) требует:
  - точный выбор – за $O(\log n)$,
  - приближенный – за $O(1)$

Результаты расчета



Результаты расчета

Среднее время ответа системы в стационарном режиме. Сравнение алгоритмов балансировки MinTime, LeastLoad, Round Robin и LeastQueue,
Система: Сервер 1 - 10 запросов/с, Сервер 2 - 2 запроса/с, Сервер 3 - 1 запрос/с.

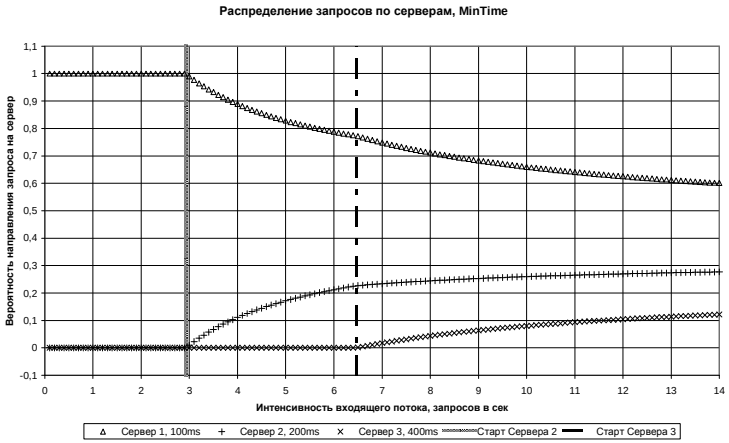


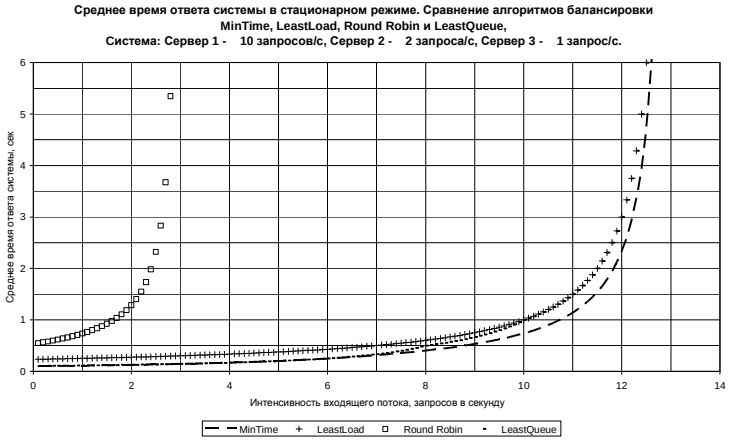
Результаты расчета

- У графиков распределения нагрузки имеются *изломы* в точках старта серверов 2 и 3

<u>Согласно расчетам:</u>

- MinTime эффективнее LeastLoad на 67% при малых нагрузках.
- MinTime эффективнее LeastLoad и LeastQueue на 26% при средних нагрузках
- MinTime эффективнее LeastLoad и LeastQueue на 23% при высоких нагрузках



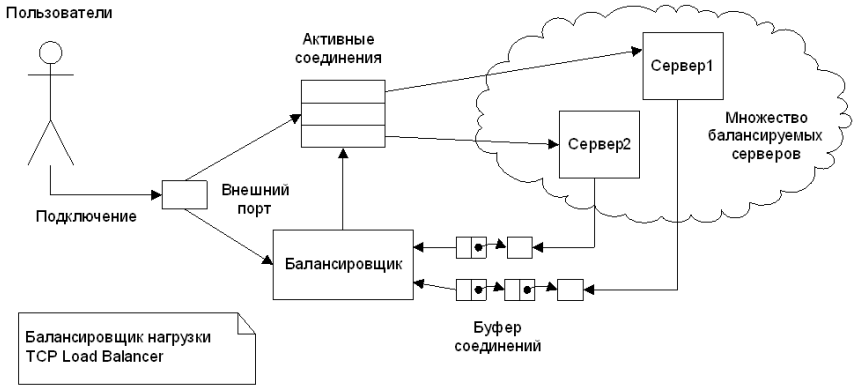


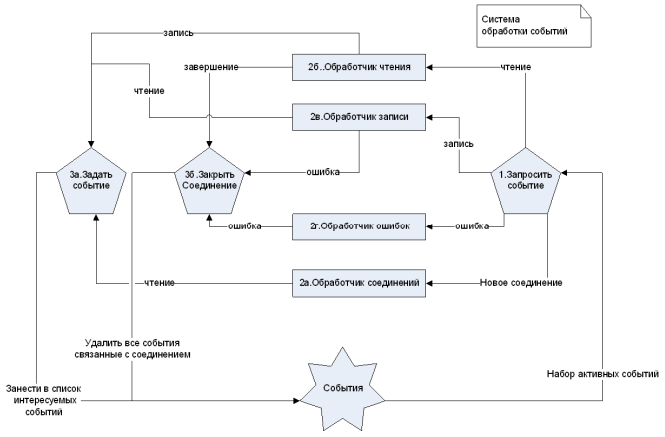
Результат

- Аналитическое решение задачи распределения нагрузки в рамках рассматриваемой модели
  - Непрерывная дифференцируемость целевой функции на оптимальном решении
  - Наличие точек старта серверов – последовательное включение серверов в работу
- Разработан алгоритм балансировки на основе полученного решения
  - Приближенный алгоритм требует постоянного от количества серверов времени для направления запроса
  - Значительный расчетный выигрыш по сравнению с существующими алгоритмами
- Применимость
  - Для любых систем обмена сообщениями и сводимых к ним
  - Ограниченность использованными предположениями

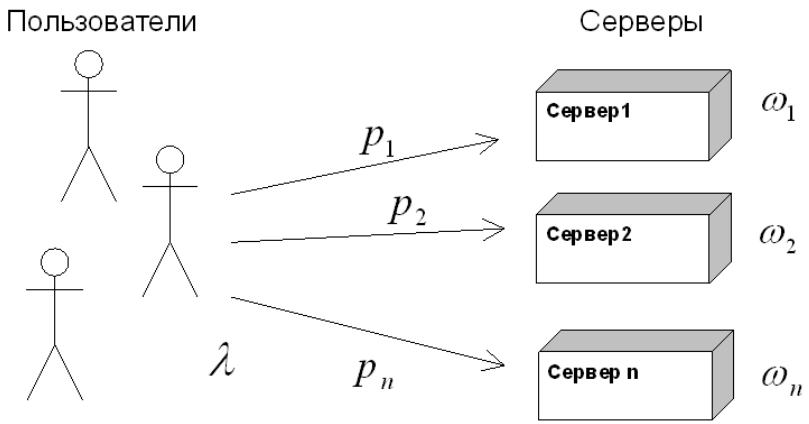
Сопутствующие проекты

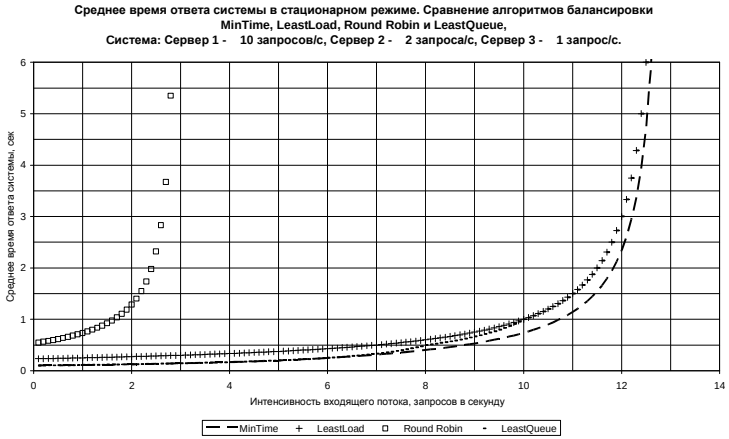
- Тестовая реализация алгоритма в TCP Load Balancer
  - Прокси на транспортном уровне стека TCP/IP
  - Событийно-управляемая архитектура
  - Переносимость на уровне кода
  - Инструмент расчета

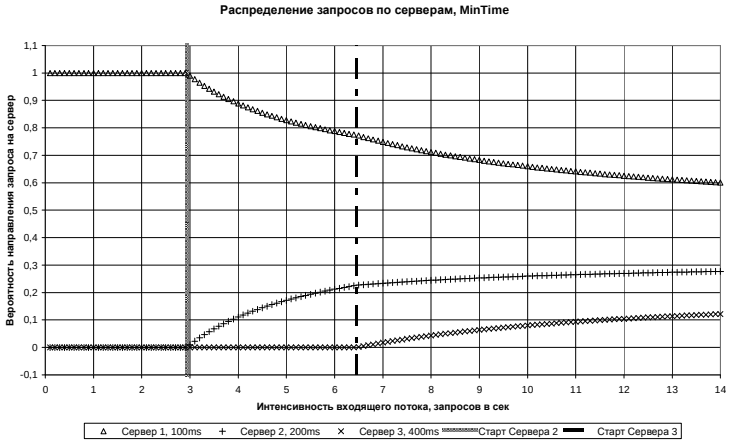


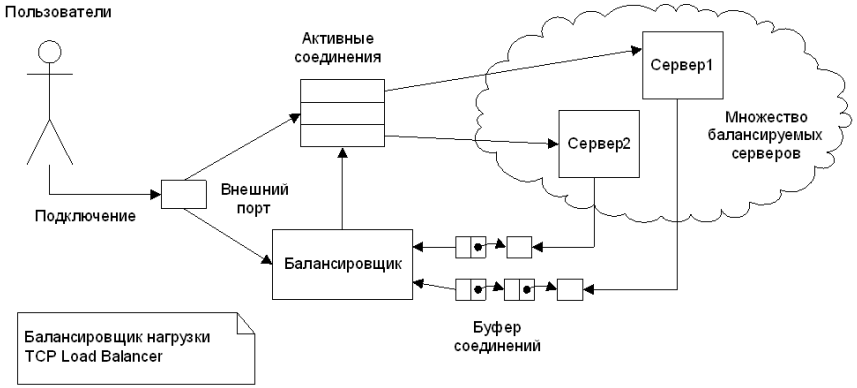


Заключение


Пользователи
Серверы
Сервер1
Сервер2
Сервер n
p_1
p_2
p_n
λ
ω_1
ω_2
ω_n


Среднее время ответа системы в стационарном режиме. Сравнение алгоритмов балансировки MinTime, LeastLoad, Round Robin и LeastQueue,
Система: Сервер 1 - 10 запросов/с, Сервер 2 - 2 запроса/с, Сервер 3 - 1 запрос/с.
Среднее время ответа системы, сек
Интенсивность входящего потока, запросов в секунду
MinTime
LeastLoad
Round Robin
LeastQueue


Распределение запросов по серверам, MinTime
Вероятность направления запроса на сервер
Интенсивность входящего потока, запросов в сек
Сервер 1, 100ms
Сервер 2, 200ms
Сервер 3, 400ms
Старт Сервера 2
Старт Сервера 3


Пользователи
Активные соединения
Сервер1
Сервер2
Множество балансируемых серверов
Внешний порт
Подключение
Балансировщик
Буфер соединений
Балансировщик нагрузки TCP Load Balancer