Оптимальное распределение нагрузки открытого типа на совокупность разнородных узлов

Хританков А.С., студент 4 курса МФТИ,
ahritankov@hotmail.ru

Апрель-Май 2005.

В данной статье получен алгоритм распределения нагрузки между серверами программно-аппаратной системы ориентированной на обработку запросов с точки зрения минимизации времени ответа системы. В качестве модели рассматривается пуассоновский входящий поток запросов, попадающий на фиксированное количество однолинейных обрабатывающих серверов. Алгоритм аналитически вычисляет оптимальный вектор, каждая из компонент которого задает вероятность направления запроса из входящего потока на соответствующий сервер системы в зависимости от интенсивности входящего потока в стационарном случае. Применение разработанного алгоритма позволяет значительно уменьшить среднее время ответа системы при малых и средних нагрузках, и полностью загрузить серверы системы при высоких нагрузках.

Ключевые слова: клиент-сервер, производительность программных систем, оптимизация производительности, теория массового обслуживания, минимальное время ответа.

1. Введение

Рассмотрим следующую модель. Пусть входящий поток запросов состоит только из запросов одного типа и является простейшим. Пусть в системе имеется n серверов, каждый из которых обрабатывает запрос в течение случайного времени ξ_i, распределенного показательно с математическим ожиданием τ_i, постоянным и известным нам точно для каждого сервера. Будем рассматривать поведение системы в стационарном случае, то есть когда время наблюдения достаточно велико и состояние системы установилось.

1.1 Формулировка задачи

Поставим задачу нахождения такого распределения запросов по серверам, при котором время ответа системы минимально. Для этого опишем модель системы в терминах теории массового обслуживания. Пусть входящий поток имеет интенсивность λ, каждый из серверов является однолинейным прибором со связанной с ним очередью. В случае если запрос поступает на сервер, занятый в этот момент обработкой другого запроса, запрос становится в очередь. Будем считать, что очередь может быть сколь угодно большой, запросы не теряются.

Из теории массового обслуживания известно, что в случае однолинейных систем, таких, как описано выше, время ответа будет выражаться формулой:

$$T_i = \delta_i + \frac{\tau_i}{1-\rho_i} \tag{1.1}$$

где δ_i – задержка коммуникации, p_i - вероятность направления запроса на сервер i,

$$\rho_i = \lambda_i \tau_i = (\lambda p_i)\tau_i \tag{1.2}$$

Время ответа i-го сервера

$$T_i = \delta_i + \frac{\tau_i}{1-\rho_i} = \delta_i + \frac{\tau_i}{1-(\lambda\rho_i)\tau_i} \tag{1.3}$$

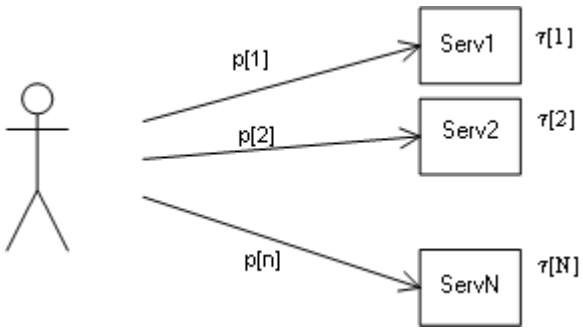

рис 1.1. Модель системы.

Среднее время ответа в таком случае равно

$$\bar{T} = \sum_{i=1}^{n} p_i T_i = \sum_{i=1}^{n} \left(p_i \delta_i + \frac{p_i \tau_i}{1-(\lambda p_i)\tau_i} \right) \tag{1.4}$$

Сформулируем поставленную задачу в новых терминах:

Найти такое распределение запросов по серверам, то есть вектор вероятностей $\{p_i\}$, что $T = T_{\min}$ по $\{p_i\}$

2. Построение алгоритма распределения нагрузки

2.1 Постановка задачи

Поставленная задача сводится к решению следующей задачи оптимизации:

$$\min_{\vec{p} \in X} F(\vec{p}), \quad F(\vec{p}) = \frac{1}{\lambda} \sum_{i=1}^{n} \left(\lambda p_i \delta_i + \frac{1}{1-(\lambda p_i)\tau_i} - 1 \right)$$

при условии, что

$$Q = \left\{ \vec{p} : \sum_{i=1}^{n} p_i = 1 \right\}, \quad P = \left\{ \vec{p} : p_i \geq 0,\ 1 - \lambda p_i \tau_i > 0,\ i = 1..n \right\}, \quad X = P \cap Q$$

причем $\delta_i \geq 0,\ \tau_i > 0,\ n \in N, \lambda \geq 0$ - известны , $i = 1..n$

(2.1)

Целевая функция *F(p)* выпукла на *P*. По определению, двойственной задачей к задаче (2.1) будет

$$\max_{y \in Y} \phi(y), \quad \phi(y) = \inf_{\vec{p} \in P} L(\vec{p}, y), \quad L(\vec{p}, y) = F(\vec{p}) + y \left(1 - \sum_{i=1}^{n} p_i \right)$$

$$Y = \left\{ y : \phi(y) > -\infty \right\}, \quad P = \left\{ \vec{p} : p_i \geq 0,\ 1 - \lambda p_i \tau_i > 0,\ i = 1..n \right\}$$

причем $\delta_i \geq 0,\ \tau_i > 0,\ n \in \square,\ \lambda \geq 0$ - известны , $i = 1..n$

$$L(\vec{p}, y) = \frac{1}{\lambda} \sum_{i=1}^{n} \left(\lambda p_i \delta_i + \frac{1}{1-(\lambda p_i)\tau_i} - 1 \right) + y \left(1 - \sum_{i=1}^{n} p_i \right)$$

(2.2)

2.2 Решение вспомогательной задачи

Найдем

$$\phi(y) = \inf_{\vec{p} \in P} L(\vec{p}, y) \tag{2.4}$$

Заметим, что функция *L* является функцией аддитивного вида и, следовательно, может быть представлена в виде

$$L(\vec{p},y)=y+\frac{1}{\lambda}\sum_{i=1}^{n}L_i(p_i,y)$$
$$L_i(p_i,y)=\lambda p_i(\delta_i-y)+\frac{1}{1-(\lambda p_i)\tau_i}-1 \qquad (2.5)$$

Понятно, что нахождение точной нижней грани для каждой из L_i по p_i даст точную нижнюю грань функции L по p, другими словами

$$\inf_{\vec{p}\in P}L(\vec{p},y)=y+\frac{1}{\lambda}\sum_{i=1}^{n}\inf_{\vec{p}_i\in P_i}L_i(p_i,y)$$
$$P_i=\{p_i:p_i\ge 0,1-\lambda p_i\tau_i>0\} \qquad (2.6)$$

P_i – выпуклое множество, L_i – строго выпуклая функция на P_i, следовательно существует единственный минимум L_i на P_i. Так как L_i дважды непрерывно дифференцируема на P_i.то этот минимум достигается либо в точке, где ее производная обращается в нуль, либо на границе P_i. При p_i стремящемся к $1/(\lambda\tau_i)$, L_i неограниченно возрастает, следовательно, минимум, так как производная L_i не меняет знака, и значение L существует на левой границе, достигается на левой границе $p_i = 0$. Первая и вторая частные производная L_i по p_i равны:

$$\frac{\partial L_i(\vec{p},y)}{\partial p_i}=\lambda(\delta_i-y)+\frac{\lambda\tau_i}{\left(1-\lambda\tau_i p_i\right)^2}=0,\quad \frac{\partial^2 L_i(\vec{p},y)}{\partial {p_i}^2}=\frac{2\left(\lambda\tau_i\right)^2}{\left(1-\lambda\tau_i p_i\right)^3}>0,\forall p_i\in P_i \qquad (2.7)$$

Вторая частная производная строго положительна, что дает строгую выпуклость функции L_i. Решая первое уравнение, получим

$$p_i^*=\frac{1}{\lambda\tau_i}\left(1-\sqrt{\frac{\tau_i}{y-\delta_i}}\right),\ \ y>\delta_i+\tau_i$$
$$p_i^*=0,\ \ y\le\delta_i+\tau_i \qquad (2.8)$$

Заметим, что при $p_i = 0$ функция L_i равняется нулю. Подставляя полученное решение обратно в L_i, найдем

$$L_i(p_i^*,y)=-\frac{y-\delta_i}{\tau_i}+2\sqrt{\frac{y-\delta_i}{\tau_i}}-1=-\left(\sqrt{\frac{y-\delta_i}{\tau_i}}-1\right)^2,\ \ y>\delta_i+\tau_i$$
$$L_i(p_i^*,y)=0,\ \ y\le\delta_i+\tau_i \qquad (2.9)$$

Подставим L_i в выражение для L и φ:

$$\inf_{\vec{p}\in P}L(\vec{p},y)=L(\vec{p}^*,y)=y-\frac{1}{\lambda}\sum_{i\in I}\left(\sqrt{\frac{y-\delta_i}{\tau_i}}-1\right)^2$$
$$I=\{i:y>\delta_i+\tau_i\} \qquad (2.10)$$
$$\phi(y)=\inf_{\vec{p}\in P}L(\vec{p},y)=y-\frac{1}{\lambda}\sum_{i\in I}\left(\sqrt{\frac{y-\delta_i}{\tau_i}}-1\right)^2$$

2.3 Решение двойственной задачи

Решим теперь задачу (2.2). Функция φ согласно теореме 3.3 (см. [1], стр. 155-156) вогнута на Y. Так как Y непусто при $n > 0$ ($y = \delta_i$ для некоторого i принадлежит Y), то согласно теореме 3.6 (см. [1], стр. 157), задача (2.2) имеет решение. Пусть серверы пронумерованы таким образом, что для всех $i > j$ выполняется $\delta_i + \tau_i > \delta_j + \tau_j$. При таком предположении условие $i \in I$ можно переписать как $i < i^*(y)$, где

$$i^*(y) = \min_{i=1..n}\{i : y \le \delta_i + \tau_i\} \tag{2.11}$$

Введем также обозначение

$$\begin{aligned} y(i^*) &= \delta_{i^*} + \tau_{i^*} \\ y(i^* - 1) &= \delta_{i^*-1} + \tau_{i^*-1} \end{aligned} \tag{2.12}$$

В данных обозначениях функции φ можно переписать при $y \in [\, y(i^* - 1),\, y(i^*)\,)$ в виде

$$\phi(y) = y - \frac{1}{\lambda}\sum_{i=1}^{i^*(y)}\left(\sqrt{\frac{y-\delta_i}{\tau_i}} - 1\right)^2, \tag{2.13}$$

$$y \in S = \left[\, y(i^* - 1),\, y(i^*)\right) \tag{2.14}$$

2.3.1 Исследование $\varphi(y)$ на максимум

Заметим, что при изменении y на S количество слагаемых в сумме постоянно, поэтому производная φ по y на S будет

$$\frac{\partial\phi(y)}{\partial y} = 1 - \frac{1}{\lambda}\omega_\Sigma^I + \frac{1}{\lambda}\sum_{i=1}^{i^*(y)}\frac{1}{\tau_i}\sqrt{\frac{\tau_i}{y-\delta_i}} = 0, \tag{2.15}$$

$$\omega_\Sigma^I = \sum_{i\in I}\frac{1}{\tau_i} = \sum_{i=1}^{i^*(y)}\frac{1}{\tau_i} = \sum_{i\in I}\omega_i, \quad \omega_i = \frac{1}{\tau_i} \tag{2.16}$$

Первое уравнение переписывается в виде

$$\sum_{i=1}^{i^*(y)}\frac{1}{\tau_i}\sqrt{\frac{\tau_i}{y-\delta_i}} = \omega_\Sigma^I - \lambda \tag{2.17}$$

Здесь необходимо сделать упрощающее предположение. Проблема состоит в том, что из последнего уравнения нужно найти y. Упрощение заключается в том, что мы будем считать, что

$$\delta_i = \delta = const, \quad \forall i = 1..n \tag{2.18}$$

Данное предположение также позволяет перейти от упорядочения по $\delta_i + \tau_i$ к упорядочению серверов по τ_i. При этом все свойства рассматриваемых функций сохранятся. Корень последнего уравнения (2.17) есть

$$\sqrt{\frac{\tau_j}{y-\delta}} = \frac{\omega_\Sigma^I - \lambda}{R_j^I} \tag{2.19}$$

$$R_j^I = \sum_{i=1}^{i^*(y)}\sqrt{\omega_i\omega_j}, \; R_\Sigma^I = \sum_{i=1}^{i^*(y)} R_j^I = \sum_{j=1}^{i^*(y)}\sum_{i=1}^{i^*(y)}\sqrt{\omega_i\omega_j} \tag{2.20}$$

$$y = \delta + \frac{R_\Sigma^I}{\left(\omega_\Sigma^I - \lambda\right)^2} \tag{2.21}$$

Полученное значение y будет лежать внутри S. Это легко показать последовательно сравнив его с верхней нижней границами промежутка S. Здесь доказательство опущено в целях экономии места. Из (2.8) и (2.19) следует важное соотношение:

$$\omega_\Sigma^I - \lambda < R_j^I \tag{2.22}$$

Это и есть условие включение индекса j во множество используемых серверов I.
Заметим, что полученное значение y единственно, так как φ строго вогнута, действительно для любых значений y:

$$\frac{\partial^2\phi(y)}{\partial y^2} = -\frac{1}{\lambda}\sum_{i=1}^{i^*(y)}\sqrt{\omega_i}\left(y-\delta\right)^{-3/2} < 0 \qquad (2.23)$$

2.3.2 Непрерывная дифференцируемость *φ(y)*

Теперь легко удостовериться в том, что серверы с большим временем обработки запроса попадают в I позднее с ростом λ, действительно, величина интенсивности входящего потока, при которой в I включается сервер с индексом i^*:

$$\lambda_{i^*} \text{ } \omega_\Sigma^I - R_{i^*}^I = \omega_\Sigma^{I-1} - \sqrt{\frac{\omega_{i^*}}{\omega_{i^*-1}}}R_{i^*-1}^I > \omega_\Sigma^{I-1} - R_{i^*-1}^I \equiv \lambda_{i^*-1}, \text{ так как } \omega_{i^*} < \omega_{i^*-1} \qquad (2.24)$$

Рассмотрим теперь случай, когда максимум φ достигается вблизи точки перехода от i^* к $i^* + 1$. Покажем, что в этом случае производная функции φ будет оставаться непрерывной как функция y, так и как функция от λ. Пусть согласно (2.11):

$$\begin{aligned} y_- &= \delta_{i^*+1} + \tau_{i^*+1} - 0 \\ y_+ &= \delta_{i^*+1} + \tau_{i^*+1} + 0 \end{aligned} \qquad (2.25)$$

Понятно, что результат операции перехода к пределу в этих выражениях есть

$$y_- = y_+ \qquad (2.26)$$

Смысл рассмотрения пределов слева и справа состоит в том, что в первом случае задействовано меньше серверов, а в сумме меньше слагаемых, чем во втором. Производная φ по y будет непрерывна, что просто получить, сравнив пределы слева и справа в точке (2.25).

$$\frac{\partial\phi(y_-)}{\partial y} = \frac{\partial\phi(y_+)}{\partial y} \qquad (2.27)$$

Аналогично показывается, что y, как функция λ, будет непрерывно дифференцируемой.

$$\frac{\partial\lambda(y_-)}{\partial y} = \frac{\partial\lambda(y_+)}{\partial y} \qquad (2.28)$$

Итак, мы получили, что производную φ можно представить как непрерывную композицию непрерывных функций от λ, значит, она непрерывна по λ.

$$\frac{\partial\phi}{\partial y}(\lambda) - \text{непрерывная функция от } \lambda \text{ для всех } \lambda \text{ для которых } X \neq \varnothing \qquad (2.29)$$

Отсюда, в частности, следует, что *значение y, задающее оптимальное решение прямо до добавления нового элемента во множество I будет давать оптимальное решение и сразу после его добавления.*

2.4 Решение исходной задачи

Подставим значение y (2.21) в выражение (2.8) для p_i , получим:

$$\begin{aligned} p_i^* &= \frac{1}{\lambda\tau_i}\left(1-\frac{\omega_\Sigma^I-\lambda}{R_i^I}\right) = \frac{\omega_i}{R_i^I} + \frac{\omega_i}{\lambda}\left(1-\frac{\omega_\Sigma^I}{R_i^I}\right), \quad R_i^I > \omega_\Sigma^I - \lambda \\ p_i^* &= 0, \quad R_i^I \le \omega_\Sigma^I - \lambda \end{aligned} \qquad (2.29)$$

$$I = \left\{i : 1 \le i \le i^*(\lambda)\right\}, \quad i^*(\lambda) = \min_{i=1..n}\left\{i : \tau_i > \frac{R_\Sigma^I}{\left(\omega_\Sigma^I-\lambda\right)^2} \Leftrightarrow \omega_i < \frac{\left(\omega_\Sigma^I-\lambda\right)^2}{R_\Sigma^I}\right\} \qquad (2.30)$$

$$R_i^I = \sum_{j\in I}\sqrt{\omega_i\omega_j}\,, \; R_\Sigma^I = \sum_{j\in I}R_j^I\,, \; \omega_\Sigma^I = \sum_{i\in I}\omega_i \qquad (2.31)$$

Полученные значения p_i задают максимум функции φ. Покажем, что они же дают минимум целевой функции F задачи (2.1). Для этого воспользуемся теоремой двойственности (см. [1], стр. 156-157 теорема 3.5). Действительно, допустимое множество X задачи (2.1):

$$X = \left\{ p_i \ge 0,\ 1 - \lambda p_i \tau_i > 0,\ i = 1..n,\ \sum_{i=1}^{n} p_i = 1 \right\}$$

непусто, например, ему принадлежит такой вектор:

$$\vec{p} = (1,\ 0\ ..\ 0) \in X \text{ при } n \ge 1,\ \ \lambda < \frac{1}{p_1 \tau_1} \tag{2.32}$$

Далее, целевая функция F задачи (2.1) всегда положительна, следовательно, и для выбранного вектора p. Это означает, что значение задачи (2.1) $F^* >= 0 > -\infty$. Таким образом, условия теоремы выполнены и выполняется соотношение двойственности (2.3). В силу того, что L достигает своей нижней грани, получим с помощью теоремы 3.6 (см. [1], стр. 157):

$$F^* = \phi^* = L(\vec{p}^*, \lambda) = \frac{1}{\lambda} \sum_{i=1}^{n} \left(\lambda p_i^* \delta_i + \frac{1}{1 - (\lambda p_i^*)\tau_i} - 1 \right) + y \left(1 - \sum_{i=1}^{n} p_i^* \right) \tag{2.33}$$

Учитывая, что последняя скобка на оптимальном решении обращается в нуль, получим для минимального времени ответа T^*, учитывая (1.4):

$$\bar{T}^* = F^* = \sum_{i=1}^{n} p_i^* \left(\delta + \frac{\tau_i}{1 - (\lambda p_i^*)\tau_i} \right) \tag{2.34}$$

Из соотношения двойственности и того факта, что на оптимальном решении φ^* является непрерывно дифференцируемой по λ на всем X следует, что T^* также непрерывно дифференцируема по λ на всем X .

2.5 Окончательный результат

Подставим в (2.34) полученные выражения для оптимального решения:

$$T^* = \delta - \frac{|I|}{\lambda} + \frac{R_\Sigma^I}{\lambda(\omega_\Sigma^I - \lambda)} = \delta + \frac{R_\Sigma^I - |I|(\omega_\Sigma^I - \lambda)}{\lambda(\omega_\Sigma^I - \lambda)},\ \ |I| = i^* \tag{2.35}$$

3. Алгоритм балансировки нагрузки MinTime

1) Рассчитать для всех $j = 1..n$ величину $1/\tau_j$ и упорядочить полученные значения по убыванию.

$$\omega_j = \frac{1}{\tau_j}$$

2) Для всех $i = 1..n$ рассчитать

$$\omega_\Sigma^i = \sum_{j=1}^{i} \omega_j\ ,\ r_\Sigma^i = \sum_{j=1}^{i} \sqrt{\omega_j}$$

3) Найти точку активации для каждого сервера по формуле, для всех $i = 1..n$

$$\lambda^i = \omega_\Sigma^{i-1} - \sqrt{\omega_i}\, r_\Sigma^i$$

4) Рассчитать для всех $i = 1..n$, $j = 1..i$

$$R_j^i = r_\Sigma^i \sqrt{\omega_j}\ ,\ \ R_\Sigma^i = \sum_{j=1}^{i} R_j^i$$

5) При заданном значении интенсивности входящего потока λ

$$p_j^* = \frac{\omega_j}{R_j^I} + \frac{\omega_j}{\lambda}\left(1 - \frac{\omega_\Sigma^I}{R_j^I}\right), \quad i = \min_{j=1..n}\left\{ j : \lambda < \lambda^j \right\},$$

при этом ожидаемое время ответа системы

$$T_{time}^* = \delta + \frac{R_\Sigma^i - i\left(\omega_\Sigma^i - \lambda\right)}{\lambda\left(\omega_\Sigma^i - \lambda\right)}$$

6) При изменении значения λ повторить с шага 5)

Замечание. При большом количестве серверов и при частых изменениях значений производительности серверов шаг 4) можно перенести в шаг 5) и вычислять только необходимые значения.

3.1 Сложность алгоритма

Шаг 1) требует *O(n log n)*, шаги 2),3) можно выполнить за *O(n)*, вычисления на шаге 4) требуют в общем случае $O(n^2)$ операций. Нахождение значения *i* на шаге 5) требует *O(log n)* операций, вычисление для каждого сервера p_i потребует в среднем *O(n)*. Остальные вычисления совершаются за *O(1)*. Поэтому при работе в устойчивом режиме при неизменных значениях производительностей серверов *алгоритм имеет логарифмическую сложность по количеству серверов (выбор сервера, на который направлять запрос)*. Была также разработана *модификация* алгоритма, которая требует *O(1)* на обработку одного запроса, другими словами, она обладает *постоянной сложностью по вычислительным ресурсам*.

3.2 Анализ

Рассмотрим систему А обработки запросов, не сохраняющую состояние. Другими словами, систему, ориентированную на обработку сообщений. Будем считать, что все запросы однотипные и требуют примерного равного времени на обработку. Пусть в распоряжении имеется три существенно различных сервера, способных обрабатывать запросы. Первый из них обрабатывает запрос в среднем за 100 миллисекунд, второй за 500 миллисекунд и третий за 1000 миллисекунд.

Сравним несколько алгоритмов балансировки: Weighted Round Robin, LeastQueue, LeastLoad и MinTime.

- Weighted Round Robin(WRR) – Распределение запросов по порядку, при котором каждый сервер обладает некоторым «весом» или приоритетом: равномерное с весами. Реализован, например, в BEA WebLogic 8.1.
- LeastQueue – Динамический алгоритм с обратной связью. Запрос будет направлен тому серверу, в очереди которого в данный момент ожидает наименьшее число запросов. Реализации неизвестны.
- LeastLoad – Динамический алгоритм с обратной связью. Запрос направляется на тот сервер, который менее всего загружен. Величина загрузки сервера может определяться, например, по времени соединения с сервером. Используется, к примеру, в Cisco Catalyst 6800.
- MinTime – Статико-динамический алгоритм без обратной связи. Распределение запросов по серверам рассчитывается для текущей нагрузки на систему. Реализован в “PoC TCP Load Balancer”, созданной автором.

Рассмотрим систему в стационарном, или установившемся, режиме, что, разумеется, не учитывает преимущества динамических алгоритмов с обратной связью, способных подстраиваться под систему. В установившемся режиме алгоритмы можно описать следующим образом: каждый из них будет стремиться к тому, чтобы выполнялось соотношение для всех *i*:

- WRR, величина $\frac{p_i}{w_i} = const,\ w_i$ - вес i-го сервера .
- LeastQueue, длина очереди на всех серверах одинакова -

  $$MQ_i = \frac{\rho_i^2}{1-\rho_i} = const,$$

  $$i = \min_{j=1..n}\left\{ j : \lambda < \lambda^j \right\}$$

  $$p_j = \frac{i \cdot \omega_j - \omega_\Sigma^i}{i \cdot \lambda} + \frac{1}{i},\ \text{при}\ \lambda > \lambda^j = \omega_\Sigma^{j-1} - (j-1)\cdot\omega_j,$$

  $$p_j = 0,\ \text{при}\ \lambda < \lambda^j$$

  $$T = \frac{i}{\omega_\Sigma^i - \lambda}$$
- LeastLoad, нагрузка на все серверы одинакова -

  $$1 - U_i = \rho_i = const,\ p_i = \frac{\omega_i}{\omega_\Sigma},\ T = \frac{n}{\omega_\Sigma - \lambda}$$
- MinTime, среднее время ответа системы минимально.

Расчет показывает, что алгоритмы LeastQueue и LeastLoad приводят к одинаковому времени ответа системы, хотя распределения запросов различны. Для алгоритма WRR можно выбрать веса таким образом, чтобы в стационарном режиме он был идентичен LeastLoad.

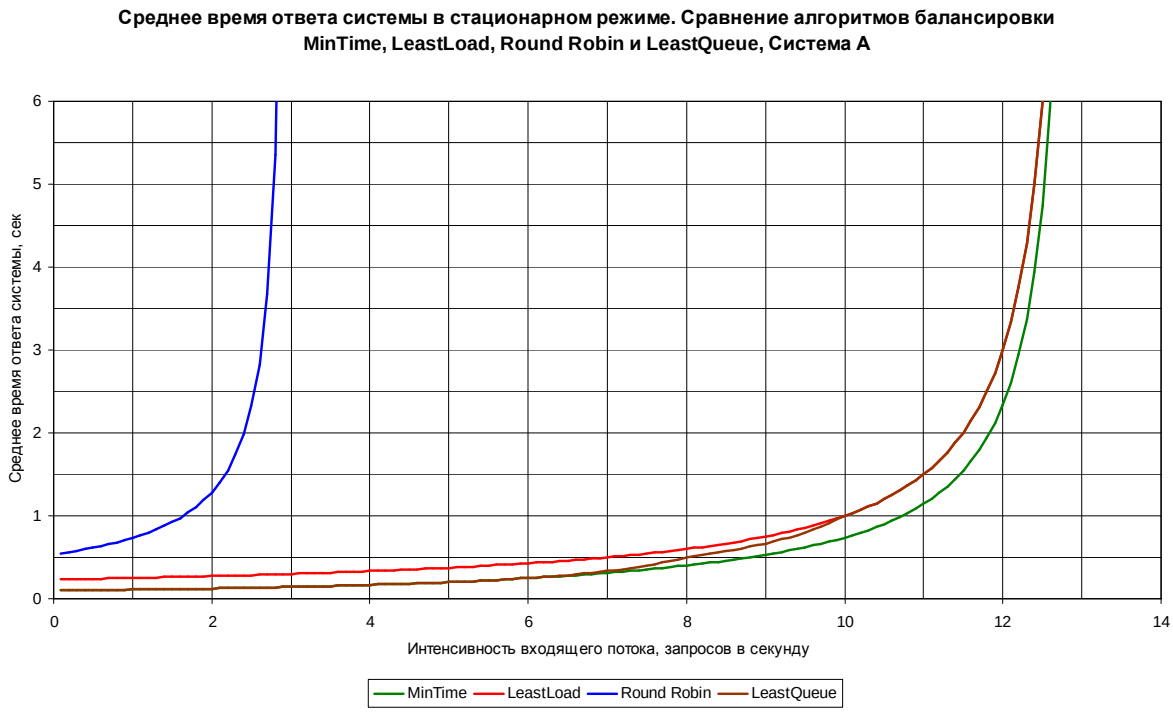

рис 3.2.1.Время ответа системы в зависимости от интенсивности входящего потока. Равномерное распределение запросов в классическом Round Robin приводит к быстрой перегрузке системы. Более оптимальные алгоритмы LeastLoad, LeastQueue и MinTime позволяют полностью загрузить систему.

рис 3.2.2. Распределение запросов по трем серверам в зависимости от интенсивности входящего потока. Синяя и коричневая кривые имеют изломы в точке старта Сервера 3

Расчет по алгоритму MinTime выполнен при помощи тестовой программы, созданной на основе “PoC TCP Load Balancer”.

Список литературы

[1] Сухарев А.Г., Тихонов А.В., Федоров В.В. «Курс методов оптимизации». М.: Наука. 1986.

[2] Б.В. Гнеденко, И.Н. Коваленко “Введение в теорию массового обслуживания”. М.: Наука.1966

Содержание